SMOOTH QUASI-NEWTON METHODS FOR NONSMOOTH OPTIMIZATION

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Jiayi Guo

May 2018

SMOOTH QUASI-NEWTON METHODS FOR NONSMOOTH OPTIMIZATION

Jiayi Guo, Ph.D.

Cornell University 2018

The success of Newton's method for smooth optimization, when Hessians are available, motivated the idea of quasi-Newton methods, which approximate Hessians in response to changes in gradients and result in superlinear convergence on smooth functions. Sporadic informal observations over several decades (and more formally in recent work of Lewis and Overton [23]) suggest that such methods also seem to work surprisingly well on nonsmooth functions. This thesis explores this phenomenon from several perspectives. First, Powell's fundamental 1976 convergence proof for the popular Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton method for smooth convex functions in fact extends to some nonsmooth settings. Secondly, removing the influence of linesearch techniques and introducing linesearch-free quasi-Newton approaches (including a version of Shor's R algorithm), shows in particular how repeated quasi-Newton updating at a single point can serve as a separation technique for convex sets. Lastly, an experimental comparison, in the nonsmooth setting, of the two most popular smooth quasi-Newton updates, BFGS and Symmetric Rank-One, emphasizes the power of the BFGS update.

BIOGRAPHICAL SKETCH

Jiayi Guo was born in Shenyang, a beautiful city located in the northeast of China. After spending the first nineteen years in his hometown, he attended the University of Illinois at Urbana-Champaign for undergraduate in 2008 and obtained an Applied Mathematics and Computer Science dual degree four years later. After graduation, he decided to pursue a doctoral degree in the School of Operations Research and Information Engineering at Cornell University.

Upon graduating from Cornell, Jiayi will join Shanghai University of Finance and Economics and begin his tenure track. At the same time, he will also work for Cardinal Operations, a start-up company in Shanghai, on designing optimization solvers and providing solutions to real industry problems.

To my parents:

Kebin Guo and Xiaoyu Feng

ACKNOWLEDGEMENTS

First and foremost, I would like to sincerely thank my adviser Adrian Lewis for his guidance, understanding, encouragement and support. Without his generous help, completion of this dissertation would not have been possible. I enjoy our weekly meeting, during which his deep knowledge guides my growth as a researcher. In addition, he has also helped me a lot in the field of academic writing and presentations. Lastly, my heart overflows with gratitude for his support of my career development by his academic resources and his open mind.

I would like to thank my committee members, Peter Frazier and David Bindel for their excellent teaching and instruction. Moreover, I also would like to thank James Renegar and David Williamson, who love to share their experience with me and has helped me in my professional development. In fact, I am very grateful to have the entire faculty and staff at the School of Operations Research and Information Engineering. My special thanks also go out to my undergraduate advisor Sheldon Jacobson, whose guidance and advices led me into the world of Operations Research.

Finally, I would like to thank my office mates Eric Ni, Angela Gao, Yixuan Zhao, Tom Fei, and Bangrui Chen, who accompanied and encouraged me through my Ph.D. study.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION AND MOTIVATION

This thesis focuses on unconstrained optimization, where we minimize an objective function with no restrictions on the values of variables. The mathematical formulation is

$$\min_x f(x)$$

where $f : \mathbf{R}^n \to \mathbf{R}$ and $x \in \mathbf{R}^n$ is a real vector with $n \geq 1$.

When the function f is convex and smooth, then various numerical methods are convergent in theory and effective in practice, examples including steepest descent, coordinate descent, trust region methods, Newton's method, and so forth [28]. If the function f is not smooth, then the steepest descent method may not converge to stationary points [19]. By contrast, the subgradient method [4], bundle methods [22], and the gradient sampling algorithm [7] are designed to solve general nonsmooth optimization problems by evaluating subgradient information at different points. In addition, Lewis and Overton [23] have explored the power of the BFGS (Broyden-Fletcher-Goldfarb-Shanno) method, a well-known quasi-Newton method (designed for smooth optimization) using gradient difference information to approximate Hessian-like matrices. They observed the surprising effectiveness of BFGS updating in nonsmooth optimization numerically and illustrated it by proving linear convergence under an exact line search in one very special case: when f is the Euclidean norm in two dimensions. Motivated by that paper, this thesis has a unified theme: to explore the reason that the BFGS method seems to work well in nonsmooth problems. Each chapter approaches this question in a different way.

If the function is convex and twice continuously differentiable, then Powell

[32] famously showed the sequence of points generated by the BFGS method with an inexact line search converges to the optimizer superlinearly, assuming a bounded initial level set $\{x : f(x) \leq f(x_0)\}$. The second chapter extends this classic theory to the nonsmooth case by slightly strengthening the convexity assumption. Notably, we prove the convergence of the BFGS method if the only nondifferentiable point is an isolated minimizer. This implies the convergence of this method if the function f is the norm function, extending the example in [23]. The content of Chapter 2 will appear as [17].

In practice, we observe that when we track iterations of BFGS, the effect of a line search complicates the analysis. Therefore, the third chapter separates the behavior of the BFGS update from line search by introducing linesearch-free BFGS algorithms. These simple algorithms seem interesting from numerical experiments. In addition, we also compare BFGS with Shor updating, another quasi-Newton-like update, in the context of convex nonsmooth optimization. To summarize, some representative examples together with some provable results provide insights into the effectiveness of BFGS updating both through the perspective of metric refinement and Cholesky factorization. The content of Chapter 3 is submitted as [18].

Lastly, in nonsmooth situations we compare BFGS numerically with the symmetric rank one (SR1) update, an alternative important quasi-Newton update. Specifically, we present a concrete example to show that an SR1 trust region method, a common method for smooth optimization, fails on nonsmooth cases. On the other hand, the SR1 method (with various modifications) using line search techniques is also tested by systematic numerical experiments. The SR1 update has a better empirical rate of convergence in the special case of norm

functions, but the BFGS method seems to be the more stable and faster algorithm, especially when problems become hard. The less effective performance of the SR1 method might be explained somehow by the rank one update itself or by the different modifications we must make. However, clearly the magic of the BFGS update in nonsmooth optimization is due to more than just simple secant or quasi-Newton properties.

CHAPTER 2

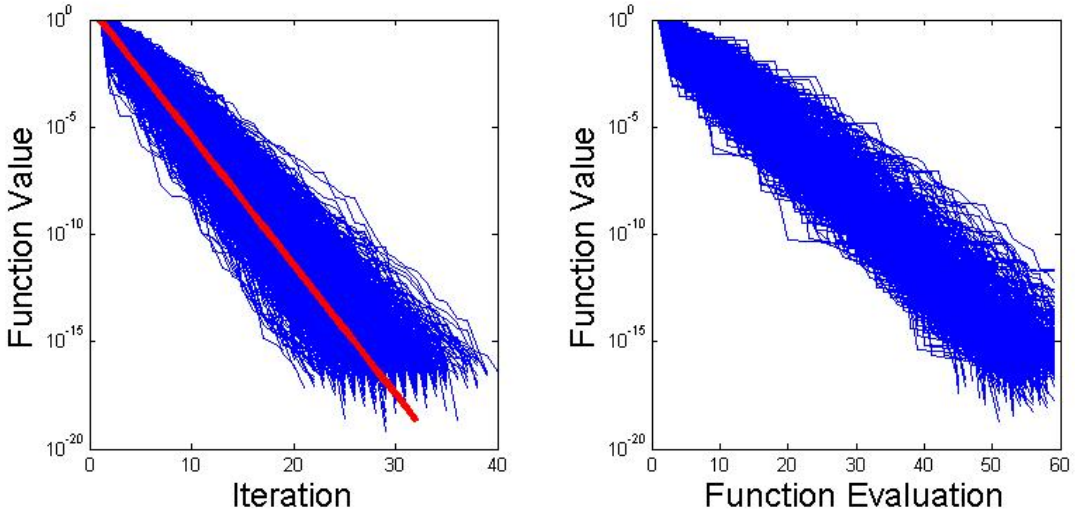
POWELL'S BFGS CONVERGENCE THEOREM UNDER NONSMOOTH ASSUMPTION

2.1 Introduction

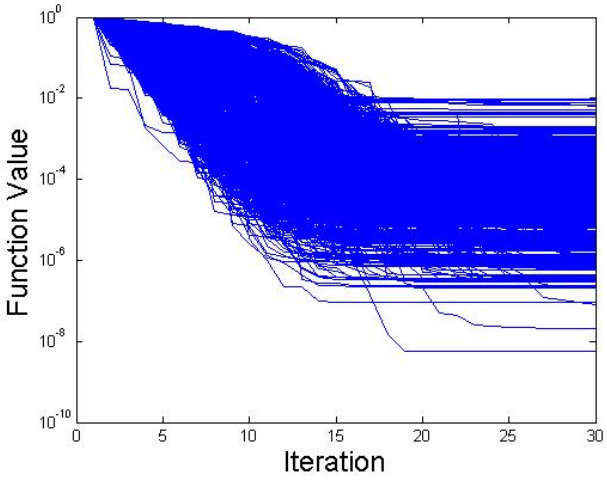
Quasi-Newton methods, which approximate the Hessian matrix using changes in gradients, has been popular for smooth optimization for decades [28]. Powell proved that the BFGS method, an example of a Quasi-Newton method, converges superlinearly on twice-differentiable convex functions with a proper line-search technique given a compact initial level set [32].

Surprisingly this method seems to work well on minimizing nonsmooth functions in practice, and in particular it is seemingly linear convergent [23]. There are very few studies of the BFGS method on nonsmooth cases. In recent work, Lewis and Overton [23] proved the linear convergence of this method on the Euclidean norm function in two dimensions under exact line search, and they conjectured that the BFGS method with inexact-line-search converges to Clarke stationary points typically.

To be more specific, we can take $f(x, y) = |x| + y^2$ as an example. A particular BFGS sequence is $(2^k, \frac{2}{5}(-1)^k 2^{-2k})$ under exact line search, the red line in the plot below. In the inexact linear search set-up, we perform a thousand runs of BFGS method with initial points sampled from a unit ball, and all sequences converge to the minimizer zero, as the figures show.



On the other hand, the steepest descent method fails to converge to the minimizer, instead converging to a nonoptimal point $(\hat{x}, 0)$ with $\hat{x} \neq 0$. The figure below indicates the behavior of this method on the function $f(x, y) = |x| + 3y^2$. (For the original function, $f(x, y) = |x| + y^2$, the iterates accidentally land and stay on the axis $y = 0$ due to an artifact of the bisection-based line search. In order to avoid this misleading performance, we choose to present the numerical result with the new function $f(x, y) = |x| + 3y^2$.) The failure of the steepest descent method for nonsmooth optimization has been known for decades: a famous example is [19, p. 363]. We provide the formal proof for our case in the next section.



Intuitively, a successful smooth algorithm should somehow detect nonsmoothness. A natural idea is to extend the convergence proof by Powell for the smooth case to nonsmooth situations. Generally speaking, if global convergence fails, then a BFGS sequence must have a subsequence of iterates converging to a nonsmooth point. Therefore, for convex functions where the minimizer is the only nonsmooth point, we can show the convergence of the BFGS method. An example is the convergence for the Euclidean norm.

2.2 Failure of steepest descent

Theorem 2.2.1. *The steepest descent algorithm with the Armijo-Wolfe line search applied to the function* $f(u, w) = u^2 + |w|$ *converges to a nonoptimal point with* $u \neq 0$ *unless* $u_k = 0$ *or* $v_k = 0$ *for some iteration* k.

Proof The proof is motivated by an analogous result [1] for the function $u + |w|$.

Suppose $x_k = (u_k, w_k) \in \mathbf{R}^2$ for $k = 0, 1, .., 2$ is a sequence of points generated by

the steepest descent method with Armijo-Wolfe line search. Specifically, there exist parameters $\mu < \nu$ in the interval $(0, 1)$ such that the vectors

$$s_k = x_{k+1} - x_k \text{ and } y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

satisfy

$$\begin{aligned} \nabla f(x_k) &\in -\mathbf{R}_+ s_k & (2.2) \\ f(x_{k+1}) &\leq f(x_k) + \mu \nabla f(x_k)^T s_k & (2.3) \\ \nabla f(x_{k+1})^T s_k &\geq \nu \nabla f(x_k)^T s_k & (2.4) \end{aligned}$$

for $k = 0, 1, 2, \ldots$. This is exactly the BFGS sequence which we introduce in the next section by replacing the Hessian updating (3.2) with identity matrices I for all iterations. Therefore, we try to prove that no such sequence converges to the minimizer, unless $u_k = 0$ or $v_k = 0$ for some iteration k.

Assume it does converge to $(0, 0)$. Without loss of generality, suppose the initial function value $f(x_0)$ is less than δ for ν and μ as Wolfe and Armijo parameters respectively. Define

$$\delta = \min\{\frac{1}{4}, \frac{1-\nu}{4\nu}, \frac{\mu}{8(1-\mu)}\}.$$

Note that given (u_k, w_k), we can check

$$(u_{k+1}, w_{k+1}) = (u_k(1 - 2t_k), w_k - t_k)$$

for some scalar $t_k > 0$. In addition, if we replace (u_k, w_k) by (u, w) for simplicity, the Wolfe condition (2.4) shows,

$$4u^2(1 - 2t - \nu) + sgn(w - t) \leq \nu.$$

Suppose $t < w < \frac{1}{2}$. Then the formula above becomes $4u^2(1 - 2t - \nu) + 1 \leq \nu \Rightarrow 1 - 4u^2\nu \leq \nu$. This contradicts the initial assumption on δ. In all, $t > w$. Moreover,

the Armijo condition (2.3) gives

$$t(1+\mu+4u^2(\mu-1+t)) \le 2w.$$

Note that

$$\begin{aligned}
& 4u^2 \ge \frac{\mu}{2(1-\mu)} \\
\Leftrightarrow\ & 4u^2(1-\mu) \ge \frac{\mu}{2} \\
\Leftrightarrow\ & 4u^2(\mu-1) \le -\frac{\mu}{2} \\
\Leftrightarrow\ & 1+\mu+4u^2(\mu-1) \le 1+\frac{\mu}{2} \\
\Leftrightarrow\ & \frac{2w}{1+\mu/2}[1+\mu+4u^2(\mu-1)] \le 2w.
\end{aligned}$$

To make the Armijo condition hold, t must be less than $\frac{2w}{1+\mu/2}$. In all, for $\gamma = \frac{2-\mu}{2+\mu} \in (0,1)$, we derive

$$w_k > 0 > w_{k+1} > -\gamma w_k.$$

By induction, $|w_k| \le \gamma^k |w_0| \to 0$ as $k \to \infty$. Furthermore, $u_{k+1} = u_k(1-2t_k)$. Therefore,

$$u_{k+1} \ge u_k(1-\frac{4|w_k|}{1+\mu/2}) \ge u_k(1-\frac{4|w_0|}{1_\mu/2}\gamma^k) \ge u_k(1-\gamma^k).$$

Therefore,

$$\log u_k \ge \log[u_0 \Pi_{j=1}^k (1-\gamma^j)] = \log u_0 + \sum_{j=1}^k \log(1-\gamma^j).$$

Since $x \to \frac{1}{x}\log(1-x)$ is a decreasing function, then we have $\log(1-\tau) \ge \frac{\tau}{\gamma}\log(1-\gamma)$ for all $0 < \tau \le \gamma$. To sum up,

$$\log u_k \ge \log u_0 + \log(1-\gamma)\sum_{j=1}^k \gamma^{j-1} \ge \log u_0 + \frac{\log(1-\gamma)}{1-\gamma}.$$

Hence, we deduce $\lim_k u_k > 0$ and have shown that the steepest descent sequence (u_k, w_k) converges to a nonzero point on the axis $w = 0$. □

2.3 BFGS sequences

Suppose that we try to minimize nonsmooth function $f : U \to \mathbf{R}$ where $U \subset \mathbf{R}^n$.

Definition 2.3.1. A sequence (x_k) is a *BFGS sequence* for the function f if f is differentiable at each iterate x_k with nonzero gradient $\nabla f(x_k)$, and there exist parameters $\mu < \nu$ in the interval $(0, 1)$ and a positive definite matrix H_0 such that the vectors

$$s_k = x_{k+1} - x_k \text{ and } y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

and the matrices defined recursively by

$$V_k = I - \frac{s_k y_k^T}{s_k^T y_k} \text{ and } H_{k+1} = V_k H_k V_k^T + \frac{s_k s_k^T}{s_k^T y_k} \tag{3.2}$$

satisfy

$$H_k \nabla f(x_k) \in -\mathbf{R}_+ s_k \tag{3.3}$$

$$f(x_{k+1}) \leq f(x_k) + \mu \nabla f(x_k)^T s_k \tag{3.4}$$

$$\nabla f(x_{k+1})^T s_k \geq \nu \nabla f(x_k)^T s_k \tag{3.5}$$

for $k = 0, 1, 2, \ldots$.

A BFGS sequence (x_k) can be generated by the standard BFGS method by a line search satisfying the Armijo-Wolfe conditions. Notice that the definition does not depend on the exact way of performing the inexact line search. Instead, only the sequence of function values $f(x_k)$ and gradients $\nabla f(x_k)$ matter.

To address this sequence more clearly, (3.2) defines the BFGS quasi-Newton updates, (3.3) shows that s_k is the approximate Newton direction, and (5.1) and (5.2) are the Armijo and Wolfe line search conditions respectively. Note that $s_k^T y_k > 0$ holds for all k, as can be proved by a simple induction.

Example: a simple nonsmooth function

Consider $f(u, v) = u^2 + |v|$. Then one possible BFGS sequence is defined by

$$\left(2^{-k}\,,\ \frac{2}{5}(-1)^k 2^{-2k}\right) \quad (k = 0, 1, 2, \ldots),$$

as observed in [24, Prop 3.2].

This sequence can also be achieved by a standard BFGS algorithm with exact line search under the following initialization.

- Starting point $\left(1, \frac{2}{5}\right)$,
- Initial Hessian $H_0 = \begin{bmatrix} \frac{1}{4} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$,
- Parameter $\mu \in (0, 0.7]$ and $\nu \in (\mu, 1)$.

2.3.1 Example: the Euclidean norm in $\mathbf{R}^2$

Consider the function $f = \|\cdot\|$ on $\mathbf{R}^2$. One possible BFGS sequence is starting at $[1\ 0]^T$, and rotating clockwise by an angle of $\frac{\pi}{3}$ and shrinking by a factor $\frac{1}{2}$ for each iteration, as observed in [23].

This sequence can also be generated by a standard BFGS algorithm with exact line search under the following initialization.

- Starting point $[1\ 0]^T$,
- Initial Hessian $H_0 = \begin{bmatrix} 3 & -\sqrt{3} \\ -\sqrt{3} & 3 \end{bmatrix}$,
- Parameter $\mu \in (0, \frac{2}{3}]$ and $\nu \in (\mu, 1)$.

2.4 Main Result

2.4.1 Review of the convergence theorem on convex smooth functions

The global convergence of BFGS is well studied by Powell for convex and smooth functions.

Theorem 2.4.1 (Powell, 1976)**.** *Consider an open convex set $U \subset \mathbf{R}^n$ containing a BFGS sequence (x_k) for a convex function $f\colon U \to \mathbf{R}$. Assume that the level set $\{x \in U : f(x) \leq f(x_0)\}$ is bounded, and that*

$$\nabla^2 f \text{ is continuous throughout } U. \tag{4.2}$$

Then the sequence of function values $f(x_k)$ converges to $\min f$.

Powell's original statement defines the function f on $\mathbf{R}^n$ instead of a convex set U. In fact, our statement is equivalent to Powell's original assumption. To see this, suppose the initial point of our BFGS sequence is x_0. Then, we can consider a level set $K = \{x : f(x) \leq f(x_0)\}$. By convexity, $K \subset U$. Moreover, f is $L-$Lipschitz on K for some L due to convexity. Define

$$\hat{f}(y) = \min_{x \in K}\{f(x) + L||y - x||\}$$

As we see, $\hat{f}$ is defined on $\mathbf{R}^n$, so Powell's theorem will be hold in this situation. Since $\hat{f}$ agrees with f on K, then $\hat{f}$ agrees with f on every point in this BFGS sequence. Therefore, our theorem follows.

Convexity is crucial in Powell's proof. In fact, the importance of convexity has been discussed by Powell himself for dimension $n > 2$ [31]. Two particular interesting nonconvex examples in [9, 26] illustrate the failure of convergence. Precisely, the first one is a polynomial $f : \mathbf{R}^4 \to \mathbf{R}$ with unbounded level sets; the second one is a C^2 smooth function with bounded level sets. On the other hand, it is still unclear whether the assumption of smoothness can be weakened or not in the convex case.

2.4.2 Motivation: a nonsmooth example

Proposition 2.4.3. *Any BFGS sequence for the Euclidean norm on $\mathbf{R}^n$ converges to zero.*

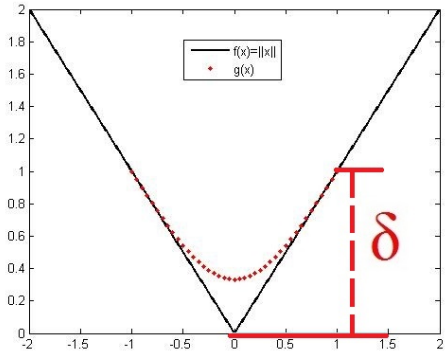
Proof

Suppose there exists a BFGS sequence $\{x_k\}$ for the Euclidean norm on $\mathbf{R}^n$ not converging to zero. Therefore, we have $\delta = \inf\{f(x_k) : k \in \mathbf{R}_+\} > 0$ with $f(x_k)$ strictly decreasing and bounded below.

Now we can find a function $g_\delta : \mathbf{R} \to \mathbf{R}$ defined by

$$g_\delta(t) = \begin{cases} \frac{\delta^3 + 3\delta t^2 - |t|^3}{3\delta^2} & (|t| \le \delta) \\ \\ |t| & (|t| \ge \delta). \end{cases} \tag{4.4}$$

Define $f_\delta(x) = g_\delta(\|x\|)$. Notice that $f(x) = f_\delta(x)$ on $\{x : f(x) \geq \delta\}$ and $f_\delta(x)$ is a convex smooth function both as a consequence of [37] and from a direct calculation. Therefore, we reach a contradiction, because Theorem 2.4.1 implies this BFGS sequence $\{x_k\}$ converges on f_δ, to the minimizer 0.



□

2.4.3 Main Theorem

The Euclidean norm provides us an insight into how Powell's Theorem for smooth optimization can apply to nonsmooth functions. As we cannot always come up with function $f_\delta(x)$ magically, a natural question is whether we can generalize this result and show the existence of such a function $f_\delta(x)$ theoretically. It turns out such generalization is possible if we strengthen the convexity assumption, but weaken the smoothness.

Theorem 2.4.5. *Powell's Theorem also holds with the smoothness assumption (4.2)*

replaced by the following assumption:

$$\begin{cases} \nabla^2 f \text{ is positive-definite and continuous throughout} \\ \text{an open set } V \subset U \text{ containing the set } \mathrm{cl}(x_k) \text{ and} \\ \text{satisfying } \inf_V f = \min f. \end{cases} \tag{4.6}$$

Proof

Without loss generality, we can consider the special case $U = \mathbf{R}^n$ and V^c is bounded.

To see this, suppose the theorem is true under the condition that $U = \mathbf{R}^n$ and V^c is bounded. For general V and U, we consider a compact level set $K = \{x \in U : f(x) \leq f(x_0)\}$ and construct $\hat{f}(y) = \min_{x \in K}\{f(x) + L\|y - x\|\}$ for all $y \in \mathbf{R}^n$ as before. In addition, for sufficient large β, $\bar{f}(x) = \max\{\hat{f}(x), \frac{1}{2}\|x\|^2 - \beta\}$ agrees with f on K. Denote $W = \{x : \hat{f}(x) < \frac{1}{2}\|x\|^2 - \beta\}$ and $\bar{V} = (V \bigcap \mathrm{int}K) \bigcup W$. Note that both W and $\bar{V}$ have bounded complement. In all, (x_k) is a BFGS sequence for both f and $\bar{f}$. Thus, the result still holds.

Now, we can safely consider just the special case where $U = \mathbf{R}^n$ and $N = V^c$ is compact. For any BFGS sequence, fix a constant $\alpha > f(x_0)$ and $\alpha > \max_N f$. Since $f(x_k)$ is decreasing and bounded below, then the closure $\mathrm{cl}(x_k)$ is also compact. Thus, $\mathrm{cl}(x_k) \bigcap N = \emptyset$ implies that the distance between these two sets is larger than a constant ϵ for some $\epsilon > 0$.

Define the distance function $d_N(x) = \min_N \|\cdot - x\|$ for $x \in \mathbf{R}^n$ and construct

$$S_\epsilon = \{x : d_N(x) \geq \epsilon \text{ and } f(x) \leq \alpha\}.$$

Note that S_ϵ is compact. In addition, we can convexify this nonconvex set S_ϵ by $\mathrm{conv}S_\epsilon$ such that a convex C^2 function f_ϵ agrees with f on S_ϵ by [39, Theorem

3.2]. Since $\{x : f(x) = \alpha\} \subset S_\epsilon \subset \{x : f(x) \leq \alpha\}$, then $\text{conv}\{x : f(x) = \alpha\} = \{x : f(x) \leq \alpha\}$.

Note that the value and gradients of the function f and f_ϵ agree for all x_k. By Theorem 2.4.1

$$f(x_k) = f_\epsilon(x_k) \to \min f_\epsilon \text{ as } k \to \infty.$$

Since $\inf_V f = \min f$, then there exists a sequence $(x^l) \in V$ such that $\lim_l f(x^l) = \min f$. To sum up,

$$\min f \leq \lim_k f(x_k) = \min f_\epsilon \leq f_\epsilon(x^r) = f(x^l) \to \min f.$$

□

We have presented an analogous result to Powell's Theorem. Specifically, the assumption of convexity is strengthened but the assumption of smoothness is weakened. In our theory, we only assume the smoothness on an open set V containing the BFGS sequence and its limit points. Note that V might exclude the minimizer. The idea of this proof is intersecting a modified version of V with a level set of f. In addition, there exists a smooth function extending f on this resulting compact nonconvex set to be convex by [39]. Therefore, we can apply Powell's theorem to this new function.

Corollary 2.4.7. *Powell's Theorem also holds with smoothness assumption (4.2) replaced by the assumption that $\nabla^2 f$ is positive-definite and continuous throughout the set $\{x \in U : f(x) > \min f\}$.*

Proof Suppose the result fails. Since $\{f(x_k)\}$ is monotonically decreasing and bounded below, then it must converge to $\min f + \epsilon$ for some $\epsilon > 0$. If we denote

$V = \{x \in U : f(x) > \min f\}$, then it contains $\mathrm{cl}(x_k)$. Therefore, Theorem 2.4.5 can be directly applied as $\inf_V f = \min f$. □

In the main result, the open set V often has full measure in the set U in practice. The case of the Euclidean norm is an example, and $f = \max_i f_i$ for strictly convex quadratics f_i is another where $V = \{x : \arg\max_i f_i(x) \text{ is unique}\}$. Then, if we randomize the initial point from a continuous probability distribution on U, then $(x_k) \subset V$ almost surely using a reasonable inexact line search to generate a BFGS sequence. Note that a precise more general argument is related to the concept of semi-algebraic functions [17]. In this way, one of the two following cases will hold.

- $f(x_k) \to \min f$
- a subsequence of (x_k) converges to a point where f is neither smooth nor minimized.

This follows the similar proof in the previous corollary. Suppose $f(x_k)$ does not converge to $\min f$. Then $f(x_k) \to \min f + \epsilon$ for some $\epsilon > 0$. Therefore, there exists a subsequence converging to $\bar{x}$ such that $f(\bar{x}) = \min f + \epsilon$. Note, by our main result, that $\bar{x}$ cannot be a smooth point.

Extensive computational experiments with BFGS suggest the first case holds almost surely [23]. However, this is not generally true for other algorithms, such as steepest descent, coordinate descent or conjugate gradients [15, 25, 28].

2.5 Constructions

Theorem 2.4.5 requires the Hessian to be positive-define on a certain set; however, some simple examples like the Euclidean norm do not satisfy this assumption. As we know, BFGS sequences actually converge for the norm function by Proposition 2.4.3. This section aims to weaken the assumption of positive-definiteness by a direct construction instead of relying on tools from [39].

Theorem 2.5.1. *Powell's Theorem also holds with the smoothness assumption (4.2) replaced by the following weaker assumption:*

$$\begin{cases} \text{For all constants } \delta > 0, \text{ there is a convex open neighborhood} \\ U_\delta \subset U \text{ of the set } \{x \in U : f(x) \le f(x_0)\}, \\ \text{and a } C^{(2)} \text{ convex function } f_\delta \colon U_\delta \to \mathbf{R} \\ \text{satisfying } f_\delta(x) = f(x) \text{ whenever } f(x_0) \ge f(x) \ge \min f + \delta. \end{cases} \tag{5.2}$$

Proof Suppose the result fails. Since $\{f(x_k)\}$ is monotonically decreasing and bounded below, then it must converge to $\min f + \epsilon$ for some $\epsilon > 0$. In addition, there exists a subsequence of (x_k) converging to $\bar{x} \in U$ such that $f(\bar{x}) = \min f + \epsilon$. In this way, we can choose $\delta = \frac{\epsilon}{2}$. The current BFGS sequence of function f is also a BFGS sequence of function f_δ. This contradicts to Theorem 2.4.1 with f replaced by f_δ. □

Theorem 2.5.3. *Powell's Theorem also holds with the smoothness assumption (4.2) replaced by the assumption that some open set $V \subset U$ containing the set* $\mathrm{cl}(x_k)$ *and satisfying* $\inf_V f = \min f$ *also satisfies the following condition:*

For any $\delta > 0$, there is a convex open neighborhood $U_\delta \subset U$ of the set

$\{x \in U : f(x) \leq f(x_0)\}$, *and a* $C^{(2)}$ *convex function* $f_\delta \colon U_\delta \to \mathbf{R}$ *satisfying* $f_\delta(x) = f(x)$ *for all* $x \in U_\delta$ *with* $d_{V^c}(x) = \min_{V^c} \|\cdot - x\| > \delta$.

Proof Denote the distance between the closed set V^c and compact set $\mathrm{cl}(x_k)$ to be $\bar{\delta}$. For all $\delta \in (0, \bar{\delta})$, $f_\delta(x_k) = f(x_k)$ due to $d_{V^c}(x) > \delta$. Therefore, the current BFGS sequence of f is also a BFGS sequence of f_δ. By Theorem 2.4.1, $f(x_k) = f_\delta(x_k) \to \min f_\delta$ as $k \to \infty$.

Since $\inf_V f = \min f$, there exists a sequence $(x^l) \in V$ such that $\lim_l x^l = \min f$. In addition, $\min f_\delta \leq f(x^l)$ for all l because $d_{V^c}(x^l) > \delta$. Therefore, $\min f_\delta \leq \min f$ implying the convergence of (x_k). □

The following result is a typical illustration.

Corollary 2.5.4. *Any BFGS sequence for the function* $f \colon \mathbf{R}^2 \to \mathbf{R}$ *defined by* $f(u, v) = u^2 + |v|$ *has a subsequence converging to a point on the line* $v = 0$.

Proof Suppose the result fails. Then, there exists a BFGS sequence $((u_k, v_k))$ with closure in $V = \{(u, v) \in \mathbf{R}^2 : v \neq 0\}$.

Note $\inf_V f = \min f$ and define $U_\delta = \mathbf{R}^2$. For all $\delta > 0$, define $f_\delta(u, v) = u^2 + g_\delta(v)$, where the function g_δ is given by equation (4.4). In all, the assumptions of Theorem 2.5.3 hold, so $f(u_k, v_k) \to 0$ implying a contradiction. Hence, the result follows. □

The numerical results at the beginning of the chapter strongly suggest that any BFGS sequence for $f(u, v) = u^2 + |v|$ in fact converges to zero. However, even in this simple case, a proof seems out of reach.

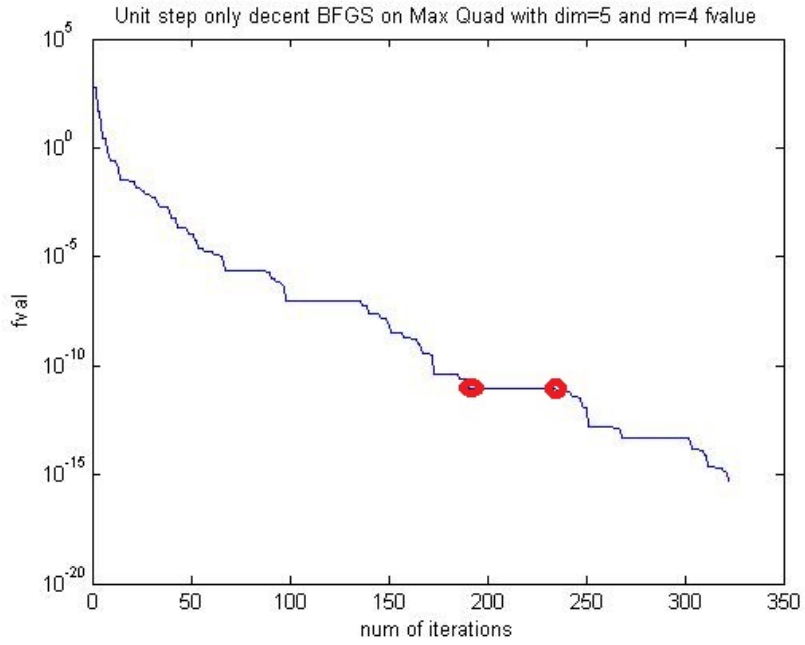
CHAPTER 3

RESCALING NONSMOOTH OPTIMIZATION USING BFGS AND SHOR UPDATES

3.1 Introduction

In the previous chapter, we have explored the power of the classical BFGS algorithm in nonsmooth optimization. It is still unclear what is the reason behind that (although the convergence is guaranteed when the optimizer is an isolated nonsmooth point). Instead of considering the BFGS matrix as approximate Hessian, we can view it as a transformation of space or "variable metric" technique. In this chapter, we compare the BFGS method with another famous variable metric method - Shor's R-algorithm in the context of convex nonsmooth optimization [38]. Shor's R-method is also a quasi-Newton-like algorithm without satisfying the secant condition [28]. Since the effect of a line search is complicated when tracked along iterations and the convergence is even more difficult to analyze when quasi-Newton-like transformations are involved, this chapter separates the behavior of such transformations from line search by creating linesearch-free algorithms. Needless to say, this simplifies the classic BFGS algorithm.

To learn whether this algorithm works or not in practice, we construct simple examples where the objective function $f : \mathbf{R}^5 \to \mathbf{R}$ is the maximum of four random strictly convex quadratics. The example is shown in the following figure, and almost always our linesearch-free BFGS algorithm converges R-linearly as this experiment indicates.



During experiments, the effectiveness of improving the local metric drew our attention. Therefore, we study a number of situations where the set of subgradients forms a polyhedron, ellipsoid or unit ball. Those representative examples together with a variety of provable results provide us some insights into the power of quasi-Newton-like updates. At the same time, this chapter introduces interesting ways of understanding such updates either through metric refinement or through Cholesky factorization. To sum up, the magic of the BFGS methodology is still not very well understood in the field of nonsmooth optimization, although this chapter gives at least some insights.

3.2 Space dilation via Shor

Recall the subgradient method takes a step from current point x in the direction $-g$ where g is a subgradient direction. Shor provided a variable metric method

which rescales the space by an $n \times n$ matrix [38], specifically

$$s = -V^T V g; \quad x_+ = x + ts.$$

In fact, we should choose an appropriate step size $t > 0$. This is not trivial in practice, though there have been promising attempts [21].

Denote a new subgradient $g_+ \in \partial f(x_+)$ and vector $e \in \mathbf{R}^n$ by

$$e = V(g_+ - g);$$

and

$$W = I - \frac{ee^T}{2\|e\|^2}; \quad V_+ = WV;$$

Therefore, the next iterate updates $x = x_+$ and $V = V_+$. This algorithm can be viewed as a steepest descent step in a transformed space [6]. Specifically, let $x = V^T y$ and $h(y) = (f \circ V^T)(y)$, and a steepest descent step of new variable y is

$$y_+ = y - t\nabla h(y).$$

Thus, we have

$$x_+ = V^T y_+ = V^T(y - t\nabla h(y)) = x - tV^T V \nabla f(x).$$

Intuitively, if the difference of consecutive gradients is huge, then W adjusts the transformed space by stretching the space along that direction.

3.2.1 Sublinear functions

Consider the special case of minimizing the sublinear function

$$f(x) = \max_{h \in Q} h^T x$$

for a given nonempty compact set $Q \in \mathbf{R}^n$. An important property of this problem is shown below.

Proposition 3.2.1. *Zero is a minimizer of f if and only if zero lies in the convex hull Q.*

Proof

- If zero is a minimizer of f, then assume 0 does not lie in the convex hull of Q. Then, there exists a hyperplane P separating 0 from Q. In addition, we can find v a vector normal to P such that $v^T h < 0$ for all $h \in Q$. This contradicts the fact that zero is a minimizer.
- If 0 lies in the interior of convex hull Q, then $0 = \sum_j \lambda_j h_j$ with $\lambda_j \geq 0$ and $\sum_j \lambda_j = 1$. Thus, $f(x) = \max_{h \in Q} h^T x \geq \max_j h_j^T x \geq \sum \lambda_j h_j^T x = 0 = f(0)$. □

Therefore, descent directions for f are normal to hyperplanes separating zero from Q. In fact, Shor's method can be applied to minimize the function f with starting point zero, and terminates once a descent direction is found. Specifically, this method always remains at $x = 0$ for all iterations, and keeps performing Shor update in order to learn the local metric information V.

In this way, one of the following two different cases will happen at each iteration.

- It terminates if $f(s) < 0$ and a descent direction is found.
- Otherwise, choose $g_+ \in \arg\max_Q \langle \cdot, s \rangle$, and set $g = g_+$ for next iteration.

The reason of this choice of g_+ may not be obvious, because we could choose g_+ to be any vector in $\partial f(s)$ theoretically. In fact, both formulations are equivalent, as the following proposition indicates.

Proposition 3.2.2. *For a sublinear function* $f : \mathbf{R}^n \to \mathbf{R}$,

$$\partial f(s) = \arg\max\{z^T s : z \in \partial f(0)\}, \quad \forall s \in \mathbf{R}^n.$$

Proof By definition in [3],

$$g \in \partial f(s) \Leftrightarrow \langle g, t - s\rangle \leq f(t) - f(s) \tag{2.3}$$

for all $t \in \mathbf{R}^n$.

Suppose $g \in \partial f(s)$. Then $-g^T s \leq -f(s)$ when $t = 0$, and $g^T s \leq f(s)$ when $t = 2s$ by (2.3). Thus, $g^T s = f(s)$. Moreover,

$$\langle g, t - s\rangle \leq f(t) - f(s) \Leftrightarrow \langle g, t\rangle \leq f(t)$$

given $g^T s = f(s)$. Therefore, $g \in \partial f(0)$ by Definition 2.3. In all,

$$\partial f(s) \subset \{z \in \partial f(0) : z^T s = f(s)\}.$$

The opposite inclusion is also easy, meaning $\partial f(s) = \{z \in \partial f(0) : z^T s = f(s)\}$. In addition,

$$\max\{z^T s : z \in \partial f(0)\} = f'(0; s) = \lim_{\epsilon \to 0} \frac{f(\epsilon s) - f(0)}{\epsilon} = \frac{\epsilon f(s) - 0}{\epsilon} = f(s).$$

Hence, the proof. □

Shor considered his algorithm as "space dilation", as explained in Section 3.2. If we define $h = Vg$ and $p = Vg_+$, we end up with the following algorithm to separate zero from a convex set.

Algorithm 3.2.4 (Shor for $0 \in \text{conv } Q$)**.**

Choose $h \in Q$; $V = I$;

while not done **do**

find a minimizer p of $\langle \cdot, h \rangle$ over Q;

if $p^T h > 0$ **then**

terminate with "$V^T p$ separates 0 from Q";

end if

$e = p - h$; $W = I - \frac{ee^T}{2\|e\|^2}$; $Q = WQ$; $V = WV$; $h = Wp$;

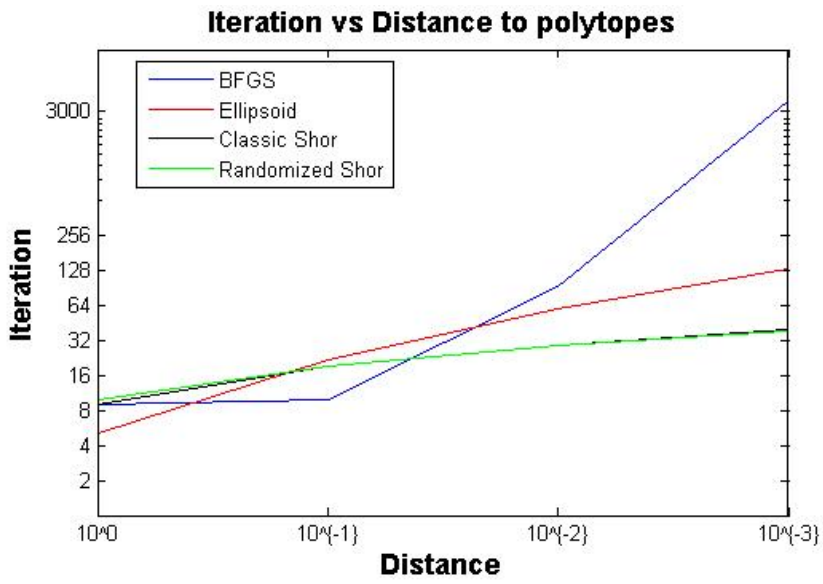
end while

Intuitively, this procedure checks whether the current h is normal to a separating hyperplane geometrically. If not, we make a simple linear transformation through W, a symmetric rank-one matrix. The simplicity of this algorithm is attractive and numerical experiments suggest the conjecture of convergence. To understand the effectiveness of Shor's R-algorithm, we present the following way to generate test examples.

1. Generate $v_i = 4^i \cdot e_i \in \mathbf{R}^5$ for all $i = 1, .., 5$ where e_i is a unit vector in each direction.
2. Choose $p = \frac{\sum 4^{-i} v_i}{\sum 4^{-i}}$.
3. Let $v_i' = v_i - (1 + \epsilon)p$. Then $\text{conv}\{v_i', -p\}$ is the desired convex set Q .

Geometrically, this Q is an irregular simplex and each v_i is a vertex of this simplex. The quantity ϵ measures the distance from point zero to this simplex. Therefore, when ϵ is small, this problem becomes ill-conditioned.

Shor's R-algorithm we introduced is "Classic Shor" in the figure, and it shows the number of iterations to terminate, averaged over random starting



points for different ϵ. From the plot, we can observe this algorithm is reliable even with $\epsilon = 10^{-3}$. The later two plots are typical trajectories of each run.

Note that this figure also compares a "Randomized Shor" with "Classic Shor" algorithm. The classic one relies on the update h and the space dilation V in a systematical way, whereas randomized Shor isolates the power of V by forgetting h after each iteration. A simple modification of the previous algorithm becomes the following.

Algorithm 3.2.5 (Randomized Shor for $0 \in \text{conv } Q$)**.**

$V = I$;

while not done **do**

choose random $u \in \mathbf{R}^n$;

find a minimizer h of $\langle \cdot, u \rangle$ over Q;

find a minimizer p of $\langle \cdot, h \rangle$ over Q;

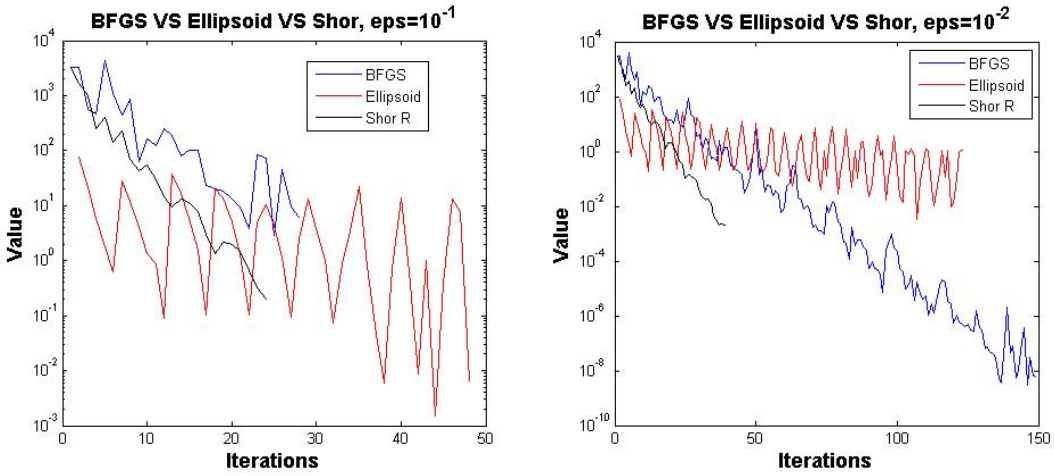
if $p^T h > 0$ **then**

terminate with $V^T p$ "normal to separating hyperplane";

end if

$e = h - p$; $W = I - \frac{ee^T}{2\|e\|^2}$; $Q = WQ$; $V = WV$;

end while



The numerical results support the benefits of V – the metric improvement clearly drives the performance of Shor's R-algorithm when it comes to minimax sublinear functions. This may not be true for separating a point from an ellipsoid, which we will introduce in the next section.

Besides Shor's R-algorithm, there are a variety of other very simple methods solving these finite minimax problems at the core of linear programming. The Perceptron algorithm and the randomized version of it are all simple and effective rescaling algorithm [12, 35]. Unlike Perceptron, Shor's R-algorithm does not preserve sparsity, though it is computationally simple [6]. The classic Perceptron algorithm takes $O(\rho^2)$ number of iterations to terminate as shown by Novikoff, where ρ is the Goffin-Cheung-Cucker condition number [29]. Note that ρ has similar scale as $\frac{1}{\epsilon}$ that we used to generate the previous ill-conditioned examples.

In addition, the performance of the Ellipsoid method can also be viewed in this figure. Actually, similarities between quasi-Newton-like algorithms and the Ellipsoid algorithm together have been discussed by many authors [2, page 1051]. Moreover, S. Bubeck derived a particular version of the Ellipsoid method similar to our set-up [5, page 249].

Algorithm 3.2.6 (Ellipsoid algorithm for $0 \in \operatorname{conv} Q$)**.**

Choose $x \in \mathbf{R}^n$, and $H \in \mathcal{S}^n_{++}$

while not done **do**

find a maximizer g of $\langle \cdot, x \rangle$ over Q;

if $g^T x < 0$ **then**

terminate with x separating 0 from $\operatorname{conv} Q$;

end if

$s - Hg$; $x = x + \frac{s}{(n+1)\sqrt{-s^T g}}$; $H = \frac{n^2}{n^2-1}(H - \frac{2ss^T}{(n+1)s^T g})$;

end while

As we see from the plot, the Ellipsoid method works, but it is not as promising as Shor's R-algorithm in terms of convergence rate. It seems neither the Ellipsoid method nor the BFGS algorithm can compete with Shor's R-algorithm from this numerical plot; however, the picture changes for the next interesting example of separating a point from ellipsoid.

3.2.2 Separating a point from an ellipsoid

What about a nonpolyhedral example? Suppose we try to separate a point c from an ellipsoid. Precisely, an ellipsoid can be treated as a unit ball $B \in \mathbf{R}^n$

transformed by an invertible $n \times n$ matrix A, and a separating hyperplane means that there exists a vector z such that

$$||A^T z|| < c^T z.$$

In our notation, we take Q to be the surface of a shifted ellipsoid,

$$Q = \{Ax - c : ||x|| = 1\}.$$

In all, Shor's R-algorithm becomes the following.

Algorithm 3.2.7 (Shor updating to separate point c from ellipsoid AB)**.**

Choose unit $x \in \mathbf{R}^n$; $V = I$;

while not done **do**

$h = Ax - c$;

$y = -A^T h$;

$y = \frac{y}{||y||}$;

$p = Ay - c$;

if $p^T h > 0$ **then**

terminate with $V^T h$ "normal to separating hyperplane";

end if

$e = h - p$; $W = I - \frac{ee^T}{2||e||^2}$; $A = WA$; $V = WV$; $c = Wc$; $x = y$;

end while

The derivation of p may follow from this proposition.

Proposition 3.2.8. *For all $h \in Q$, choose $y = \frac{A^T h}{||A^T h||}$. Then $y = \arg\max_{y \in B} \langle A \cdot -c, h \rangle$.*

Proof

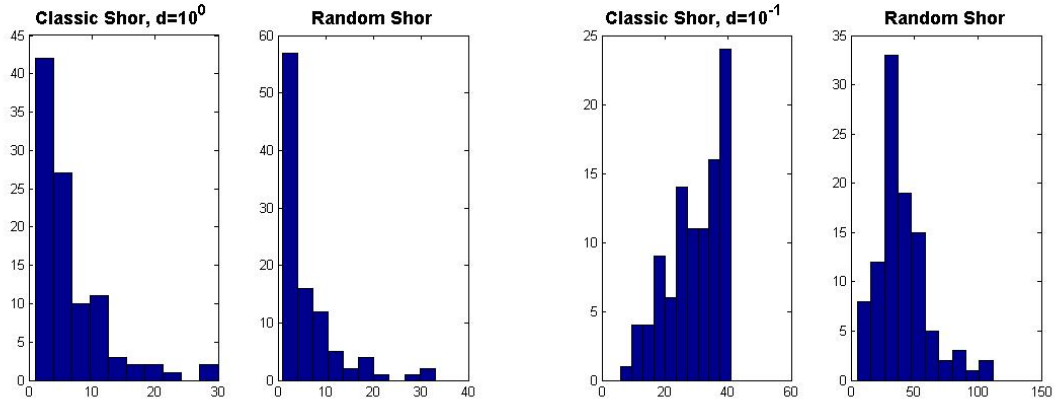
$$\begin{aligned} \arg\max_{y \in B} \langle A \cdot -c, h \rangle &= \arg\max_{y \in B} \{h^T Ay - h^T c\} \\ &= \arg\max_{y \in B} h^T Ay \end{aligned}$$

Therefore, optimal y must be in direction of $A^T h$. Hence, this proposition. □

We construct an ill-conditioned example as usual in order to understand the performance of these algorithms. Specifically, consider $n = 5$ and a diagonal matrix A =diag(1, 10, 10^2, 10^3, 10^4). A hundred instances are generated by setting $c = (1+d)Au$ for some random unit vectors $u \in \mathbf{R}^5$. As we see, d measures how ill-conditioned this problem is.

A comparison between "Classic Shor" and "Randomized Shor" is outlined in these figures. To define the randomized algorithm, we replace $x = y$ by:

1. Fix a random $u \in \mathbf{R}^n$;
2. $x = A^T u$;
3. $x = \frac{x}{\|x\|}$;



The numerical experiment indicates that Shor's R-algorithm can separate a low-dimensional ellipsoid, moderately ill-conditioned, from zero in dozens of iterations typically. Unsurprisingly, the total number of iterations required to terminate grows as d shrinks.

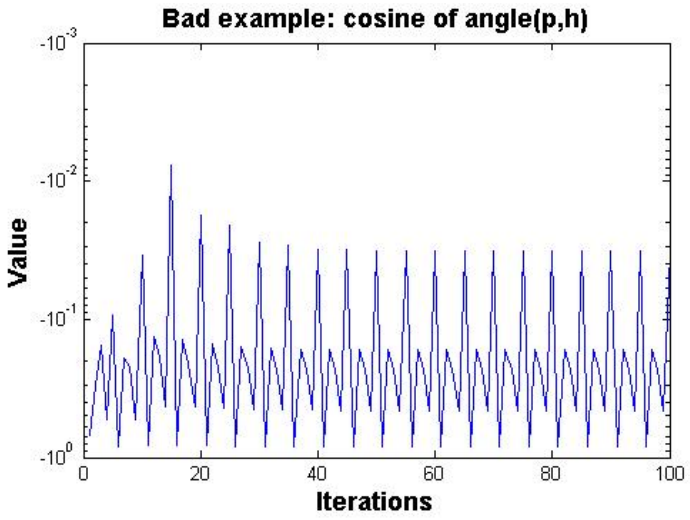
When $d = 10^{-2}$, the classic algorithm takes around a hundred iterations whereas the randomized one terminates in thousands. Therefore, it seems helpful if we reuse the information of the previous vector in Q.

As the classical theory says little, we would like to prove convergence of Shor's R-algorithm even in a special case. However, we know little more than what other authors achieved [6].

At the same time, we found the classic algorithm can *fail* occasionally. For example, consider the following

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix}, \quad v = -\begin{bmatrix} 10 \\ 39 \end{bmatrix}, \quad c = (1 + 10^{-2})A\frac{v}{\|v\|}.$$

The algorithm does not terminate in even a thousand iterations. Furthermore, a cyclical behavior with a period of five iterations can be seen from the plot. In particular, $\frac{p^T h}{\|p\|\|h\|} < -\frac{1}{100}$ eventually, which clearly indicates the failure of the termination criterion.



Interestingly, we observe no failure for the randomized algorithm. It seems

randomization can prevent bad performance which might be due to the set-up of algorithm itself.

In the context of a randomized algorithm, another number we can randomize is the factor 2 that appears in the denominator of $W = I - \frac{ee^T}{2\|e\|^2}$. It can be replaced by any constant greater than 1, served as a "learning rate", especially since $\det(W, \gamma) = \det(I - \frac{ee^T}{\gamma\|e\|^2}) = 1 - \frac{1}{\gamma}$. One might conjecture that numerical failure will not occur for the "Classic Shor" algorithm even if we only randomize this learning rate.

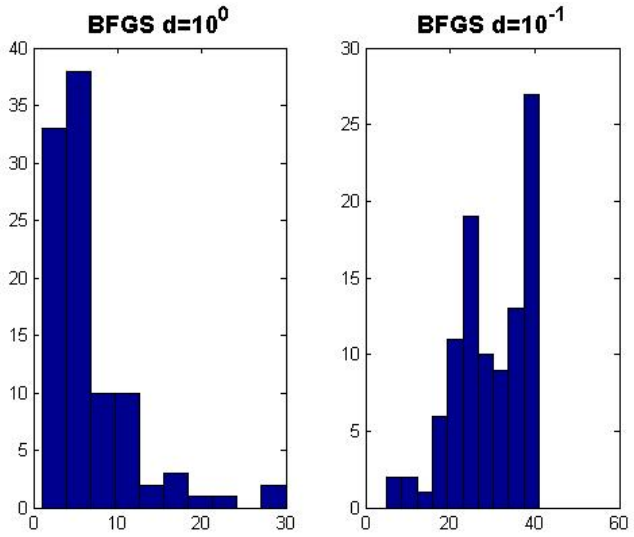
We have noted several times in this thesis that the BFGS method, as a general-purpose nonsmooth optimization algorithm, performs successfully in practice [23]. It seems more promising than Shor's R-algorithm. So as to compare with Shor's R-algorithm, we can replace the formulation of W

$$W = I - \frac{ee^T}{2\|e\|^2}$$

by the BFGS version:

$$W = I - \Big(\frac{e}{h^T e} - \frac{h}{\|h\|\sqrt{h^T e}}\Big)h^T$$

under the same set-up (The derivation of new W will be shown in Section 3.7 in detail). As we see, both W are rank one perturbation of identity; however, the latter one is not symmetric.



In all, BFGS method works, but seems slower than Shor's R-algorithm as the polyhedral becomes more and more ill-conditioned. At the same time, the BFGS algorithm performs almost the same as Shor's R-algorithm for ellipsoid separation examples, as it is illustrated by the figure above, though "Classic Shor" algorithm may fail occasionally.

3.3 The BFGS update

Before going deeper, we first recall BFGS algorithm as a steepest descent method iteratively approximating a local metric.

Suppose we minimize an unconstrained convex smooth function $f \in C^2$, and for all $x \in \mathbf{R}^n$, $\nabla^2 f(x)$ is positive definite. If we denote $g = \nabla f(x)$ at the current point x, then

- Gradient descent method: the local metric corresponds to the unit ball.

Then the unit step $s = \arg\max\{g^T s : \|s\| \leq 1\} = -\frac{g}{\|g\|}$.

- Newton's method: the local metric now corresponds to an ellipsoid. Then $s = \arg\max\{g^T s : s^T \nabla^2 f(x) s \leq 1\} = -\nabla^2 f(x)^{-1} g$.

We typically make the update $x_+ = x + ts$, and t can be found by standard backtracking line search. Basically, Newton's method accelerates the procedure through a more accurate local metric information. When the current point is close to minimizer, a unit step will always be acceptable by Newton's method.

When it comes to the BFGS algorithm, the idea is to replace the inverse of the true Hessian matrix $\nabla^2 f(x)^{-1}$ by an approximation H. The positive definite matrix H will update along iterations and hopefully $s = -Hg$ is close enough to $-\nabla^2 f(x)^{-1} g$ to approximate Newton's method.

Specifically, the Hessian update can be seen as a $S^n_{++} \to S^n_{++}$ map

$$\text{BFGS}_{f,x,t}(H) = H_+$$

as follows:

$$\begin{aligned} s = -Hg, \quad x_+ = x + ts, \\ g_+ = \nabla f(x_+), \quad y = g_+ - g, \\ V = I - \frac{sy^T}{s^T y}, H_+ = VHV^T + \frac{ss^T}{s^T y} \end{aligned}$$

assuming $s^T y > 0$.

Then, the BFGS algorithm updates

$$x = x_+; \; g = g_+; \; H = H_+$$

for the next iteration. Using a line search, Newton method is eventually quadratically convergent whereas the BFGS algorithm is superlinearly convergent [28].

Surprisingly, the BFGS algorithm, from smooth optimization, is also very effective for nonsmooth optimization [23]. Of course, one may wonder whether this algorithm is well-defined where there exist nondifferentiable points. However, if we assume that the space of nonsmooth points has a strictly lower dimension than $\mathbf{R}^n$ (polyhedral functions are a simple example), then the probability of hitting nonsmooth points is zero with a suitably randomized initial point [23].

Consider $f(x, y) = |x| + y^2$. The nonsmooth points lies in a line $x = 0$, one dimensional space in $\mathbf{R}^2$. The BFGS algorithm is almost surely well-defined under a reasonable inexact line search, like bisection backtracking, and a suitably randomized starting point, say normally distributed.

The line search always complicates the analysis of BFGS algorithm, especially if we would like to focus on the Hessian update itself. As we prefer to downplay the line search, an idea of $unit - step\ BFGS$ algorithm emerges by choosing $t = 1$ for all iterations. If we meet a "bad" point using the unit step, meaning $f(x_+) \geq f(x)$, then we will update H but stay at the current point x. Thus, we end up with the following algorithm:

Algorithm 3.3.1 (BFGS updating)**.**

Input: x, H, and set $g = \nabla f(x)$

while $f(x - Hg) \geq f(x)$ **do**

$H = \text{BFGS}_{f,x,1}(H)$;

end while

Output: $x_+ = x - Hg$, $H_+ = H$

Using Algorithm 3.3.1 as an oracle, we have

Algorithm 3.3.2 (Linesearch-free BFGS)**.**

while not done **do**

(x, H) = Algorithm 3.3.1(x, H)

end while

To make Algorithm 3.3.2 well defined, we must require the function f to be strictly convex. In this way, $s^T y > 0$ is guaranteed for all iterations if we choose the initial Hessian to be positive definite. By similar analysis, this algorithm also makes sense for nonsmooth optimization under reasonable assumptions. It seems that this attempt at trying line search free BFGS on nonsmooth optimization is new. Surprisingly, this algorithm seems quite promising, as indicated by the figure in the introduction to this chapter.

Various numerical experiments suggest the following intuition. If the current point is close to the optimizer, then Algorithm 3.3.2 is the same as the BFGS algorithm with line search, because, like Newton's method, the unit step will always be acceptable and ensure convergence [28]. If the current point is far from the optimizer, then this algorithm will keep learning curvature information through approximating the inverse Hessian matrix H even when $f(x - Hg) \geq f(x)$, and hopefully we can end up with some x_+ with $f(x_+) < f(x)$ in finitely many steps. When it comes to smooth optimization, the algorithm might work, because the assumption of strict convexity is strong. However, it is another story in nonsmooth optimization. Moreover, one concern is the num-

ber of iterations before moving to the next point. However, a line search may require a lot of function evaluations within each iteration, too [23]. Therefore, this algorithm may be no worse than or even better than the classic one in terms of total number of function evaluations.

3.4 An exploration of the linesearch-free BFGS algorithm

After this discussion, we may ask a list of questions to ourselves and hope the rest of the chapter can provide meaningful insights.

- What does the unit step BFGS update - $\text{BFGS}_{f,x,1}(H)$ tell us?
- Does Algorithm 3.3.1 always terminate?
- What does this algorithm mean when the starting point is not differentiable?
- Could the unit step BFGS update underly the magic of BFGS in nonsmooth optimization?

3.4.1 Smooth functions

In the section, we consider the objective function to be twice differentiable functions. When the current point x is close enough to optimal solution, the BFGS iterate with unit step can always generate a descent point [28], so the loop in Algorithm 3.3.1 is only performed once. As a result, Algorithm 3.3.2 converges linearly in a local region, if the Hessian matrix is locally Lipschitz continuous [10,

Thm 7.6]. However, whether this algorithm converges globally is still unknown. Therefore, this section mainly explores the property of global convergence.

Consider the simplest case when a strictly convex function $f \in C^1$ is univariate.

Theorem 3.4.1. *For any C^1 smooth strictly convex function $f : \mathbf{R} \to \mathbf{R}$ at any noncritical point x, BFGS updating (Algorithm 3.3.1) terminates in a finite number of steps.*

Proof Suppose the iteration does not terminate.

Note that the positive definite matrix H simply becomes a scalar $h > 0$. Without loss of generality, assume $g = f'(x) = 1$.

In this way, $s = -h$, $x_+ = x - h$, $g_+ = f'(x-h)$, $y = f'(x-h) - 1$, $V = 0$, and

$$h \leftarrow h_+ = \frac{h}{1 - f'(x-h)}.$$

By the definition of strict convexity,

$$f(x) \geq f(x-h) + f'(x-h) \cdot h + \frac{m}{2}h^2$$

for some $m > 0$, and this is equivalent to

$$f'(x-h) \leq \frac{f(x) - f(x-h) - \frac{m}{2}h^2}{h}.$$

Since $f(x-h) \geq f(x)$ by assumption this implies $f'(x-h) \leq -\epsilon < 0$ for some $\epsilon > 0$.

Hence $h_+ = \frac{h}{1-f'(x-h)} \leq \frac{h}{1+\epsilon}$ at every iteration. Therefore, x_+ could be arbitrarily close to x. Since f' is a continuous function, then there exists some x_+ falling into the region $\{x : 0.5 < f'(x) < 1\}$.

From Taylor series, $f(x) \geq f(x_+) + hf'(x_+)$ given $f''(x_+) > 0$. In other words, $f(x_+) \leq f(x) - hf'(x_+) < f(x)$. This completes the proof by contradiction. □

Next we will study the behavior of this algorithm for multivariate quadratic functions. Before reaching that, consider the special case $f(x) = \frac{1}{2}\|x\|^2$ first. Accordingly, Algorithm 3.3.1 becomes the following.

Algorithm 3.4.2 (BFGS updating for $\frac{1}{2}\|\cdot\|^2$)**.**

while $\|x - Hx\| \geq \|x\|$ **do**

$s = -Hx$;

$z = \frac{s}{\|s\|}$;

$H = (I - zz^T)H(I - zz^T) + zz^T$;

end while

In fact, the algorithm is similar if we choose $f(x) = \frac{1}{2}\|Rx\|^2$, a general strictly convex quadratic function for any R invertible. We simply replace $\hat{x} = Rx$ and $\hat{H} = RHR^T$ in the above algorithm.

Before we proceed, define the inner product of two symmetric matrices in S^n by $\langle X, Y\rangle$ =trace(XY), and matrix norm by $\|X\| = \sqrt{\langle X, X\rangle}$. In fact, this inner product is useful to derive standard theory from the classical quasi-Newton literature [10, 16].

Lemma 3.4.3. *Suppose we have a symmetric matrix $P \in S^n$ with $P^2 = P$. Then for all $X \in S^n$, the matrix $X_+ = PXP$ is orthogonal to the matrix $X_+ - X$.*

Proof

$$\begin{aligned}\text{trace}(X_+(X_+ - X)) &= \text{trace}(PXP(PXP - X)) = \text{trace}(PXPPXP - PXPX)\\ &= \text{trace}(PXPXP - PXPX) = \text{trace}(PXPX(P - I))\\ &= \text{trace}(XPX(P - I)P) = \text{trace}(XPX(P^2 - P)) = 0. \quad \square\end{aligned}$$

Furthermore, let $\lambda_{min}(H)$ be the smallest eigenvalue of matrix $H \in S^n$.

Lemma 3.4.4. *For any unit vector $z \in \mathbf{R}^n$, and any matrix $H \in S^n$, the matrix*

$$H_+ = (I - zz^T)H(I - zz^T) + zz^T$$

satisfies

$$H_+ - I \ \perp \ H_+ - H, \tag{4.5}$$

and consequently

$$\|(H - I)z\|^2 \ \leq \ \|H - I\|^2 - \|H_+ - I\|^2, \tag{4.6}$$

and furthermore

$$\lambda_{min}(H_+) \ \geq \ \min\{\lambda_{min}(H), 1\}. \tag{4.7}$$

Proof Let $P = I - zz^T$ and $X = H - I$. Then, as z is a unit vector,

$$H_+ - I = (I - zz^T)H(I - zz^T) - (I - zz^T) = (I - zz^T)(H - I)(I - zz^T) = X_+,$$

which proves (4.5) by the previous lemma. Furthermore, we have

$$H_+z = (I - zz^T)H(I - zz^T)z + zz^Tz = (I - zz^T)H(z - z) + z = z.$$

Therefore, (4.6) can be derived as follows:

$$\begin{aligned}
\|(H - I)z\|^2 &= \|(H - H_+)z\|^2 && \text{(by } H_+z = z\text{)} \\
&\leq \|H - H_+\|^2 && \text{(by } \|z\| = 1\text{)} \\
&= \|H - I\|^2 - \|H_+ - I\|^2 && \text{(by 4.5).}
\end{aligned}$$

In addition, suppose a unit vector $u \in \mathbf{R}^n$ satisfies $\lambda_{min}(H_+) = u^T H_+ u$. Thus,

$$\begin{aligned}
\lambda_{\min}(H_+) &= (u - (z^Tu)z)^T H(u - (z^Tu)z) + (z^Tu)^2 \\
&\geq \|u - (z^Tu)z\|^2 \lambda_{\min}(H) + (z^Tu)^2 \\
&= (1 - (z^Tu)^2)\lambda_{\min}(H) + (z^Tu)^2 \\
&= (1 - \lambda_{\min}(H))(z^Tu)^2 + \lambda_{\min}(H).
\end{aligned}$$

which implies (4.7). □

Theorem 3.4.8. *Algorithm 3.3.1 terminates for any strictly convex quadratic function f if the starting point is not the minimizer.*

Proof By (4.6), $\|H - I\|$ is nonincreasing. Since $\|X\|^2 = \sum_i \lambda_i^2(X)$ where λ_i is an eigenvalue of matrix, then the sequence $\lambda_{\max}(H-I)$ is bounded, and so is $\lambda_{\max}(H)$. Thus, $\|H-I\|$ is nonincreasing and bounded by zero, so this sequence converges. In other word, $(H - I)z \to 0$.

By (4.7), $\lambda_{\min}(H) \geq \min\{\lambda_{\min}(H_0), 1\} > 0$. Thus, the matrix H^{-1} is uniformly bounded. In all,

$$\frac{s+x}{\|s\|} = \frac{s - H^{-1}s}{\|s\|} = (I - H^{-1})\frac{s}{\|s\|} = (I - H^{-1})z = H^{-1}(H - I)z \to 0.$$

In all, $s \to -x$, meaning $x_+ = x + s \to 0$.

As $f(x) > 0$ due to the assumption that the current point x is not a stationary point and $f(x_+) \to f(0) = 0$, then Algorithm 3.3.1 terminates eventually. □

Furthermore, this proof also tells us about convergence for the linesearch-free BFGS algorithm 3.3.2.

Theorem 3.4.9. *The linesearch-free BFGS Algorithm* 3.3.2 *converges superlinearly to the minimizer of any strictly convex quadratic function.*

Proof If we calculate a new point x_+ from the current point x, the BFGS update is always performed regardless of the comparison between $f(x_+)$ and $f(x)$ in

Algorithm 3.3.2. Therefore, by the same analysis from the previous theorem - only depending on a sequence of BFGS updates, we have

$$\frac{x_+}{\|s\|} = \frac{s+x}{\|s\|} \to 0.$$

In addition, we know that $\{x_k\}$ has bounded norm by the assumption of convex quadratic functions, and H has bounded norm, so $\|s\| = \|Hx\|$ is bounded. Therefore,

$$\frac{x_+}{\|s\|} \to 0 \text{ implies } x_+ \to 0.$$

In fact, we also know

$$\begin{aligned} s \to -x &\Leftrightarrow s^T s \to -s^T x \\ &\Leftrightarrow \frac{s^T x}{\|s\|^2} \to -1 \\ &\Leftrightarrow \frac{\|x_+\|^2 - \|x\|^2}{\|s\|^2} \to -1. \end{aligned}$$

Thus, eventually the algorithm will accept the unit step due to $\|x_+\| < \|x\|$. In this way, we can reduce linesearch-free BFGS to the classic algorithm leading to superlinear convergence [28]. □

Powell's theory [32] indicates the convergence of the BFGS algorithm on convex and smooth functions with a proper line-search technique given a compact initial level set. Particularly, when the current point is close enough to the optimizer, the unit step will always be accepted by an Armijo-Wolfe line search with the Wolfe conditions (2.4) with $\nu \leq \frac{1}{2}$ [28]. Therefore, we cannot distinguish between the classic algorithm and our linesearch-free algorithm locally in this setting. Furthermore, the fact that the linesearch-free BFGS algorithm converges on convex quadratics implies that we may possibly extend the theory of convergence to more general problems, so we reach the following conjecture.

Conjecture 3.4.10. *Linesearch-free BFGS Algorithm 3.3.2 converges if function f is strongly convex and smooth given a compact initial level set.*

Note that we know there exist pathological counterexamples to convergence if we drop the assumption of convexity, even when the function we optimized is smooth [9, 26]. Thus, the convexity assumption is critical.

3.4.2 Nonsmooth functions

From the discussion in Section 3.3, we know this algorithm can be applied even to nonsmooth functions. The figure in Section 3.1 is an example of the typical effectiveness of this algorithm. However, progress in proving the convergence of Algorithm 3.3.2 is slight. Before we can say anything about convergence, the termination of Algorithm 3.3.1 needs to be understood. We don't know whether it is true even for smooth cases, though a number of numerical results suggest so.

3.5 BFGS updating for nonsmooth functions

Exploration suggests that Algorithm 3.3.1 is also suited for the case when the current point x is not differentiable. To understand how the BFGS update oracle can be useful, let's define it as the following.

Consider a convex nonsmooth function $f : \mathbf{R}^n \to \mathbf{R}$, a nondifferentiable current point x, a current subgradient $g = \partial f(x)$, and a current matrix $H \in S^n_{++}$.

Algorithm 3.5.1 (BFGS updating - nonsmooth)**.**

while true **do**

$s - Hg, x_+ = x + s$

$g_+ \in \arg\max\{z^T s : z \in \partial f(x_+)\}$

$y = g_+ - g$, $V = I - \frac{sy^T}{s^T y}$, $H_+ = VHV^T + \frac{ss^T}{s^T y}$

$g = \arg\max\{z^T s : z \in \partial f(x)\}$, $H = H_+$

end while

The choice of g and g_+ and updates follow exactly as discussed in Section 4.2.

To make sure this algorithm is effective, we need to answer two questions:

- If x is not a minimizer, will Algorithm 3.5.1 terminate?
- If x is a minimizer, will the step $-Hg$ converge to zero?

3.5.1 Sublinear functions

Suppose the starting point is 0. Denote $C = \partial f(0)$ where f is a sublinear function. Thus, Algorithm 3.5.1 becomes

Algorithm 3.5.2 (BFGS for $0 \in C$)**.**

Choose $g \in C$, and $H \in S^n_{++}$;

for $k = 0, 1, 2, \ldots$ **do**

$s = -Hg$;

Find a maximizer g_+ of $\langle \cdot, s \rangle$ over D;

if $g_+^T s < 0$ **then**

terminate with "s normal to separating hyperplane ";

end if

$y = g_+ - g$; $V = I - \frac{sy^T}{s^T y}$; $H = VHV^T + \frac{ss^T}{s^T y}$; $g = g_+$;

end for

The algorithm is well-defined, because if both stopping criteria fail, then

$$y^T s = g_+^T s - g^T s = g_+^T s + g^T H g > 0$$

given $g_+^T s \geq 0$ and $g \neq 0$.

If $C = \text{conv} D$, the algorithm can be simplified, because we can compute $g_+ \in D$ rather than $g_+ \in C$. Therefore, we arrive at the following algorithm.

Algorithm 3.5.3 (BFGS for $0 \in \text{conv } D$)**.**

Choose $g \in D$, and $H \in S_{++}^n$;

for $k = 0, 1, 2, \ldots$ **do**

$s = -Hg$;

Find a maximizer g_+ of $\langle \cdot, s \rangle$ over D;

if $g_+^T s < 0$ **then**

terminate with "s separates 0 from conv D";

end if

$y = g_+ - g$; $V = I - \frac{sy^T}{s^T y}$; $H = VHV^T + \frac{ss^T}{s^T y}$; $g = g_+$;

end for

Having this set-up, we still need to explain the two immediate questions:

- $0 \notin C \Rightarrow$ termination in finite iterations?
- $0 \in C \Rightarrow$ either termination or step s converges to zero?

3.6 Symmetry and the unit ball

Let's consider the second question where $0 \in C$. Define the following measure

$$\text{sym}(C) = \max\{t : g \in C \Rightarrow -tg \in C\}$$

The measure is also introduced elsewhere in the literature when it comes to complexity analysis [14, 27, 38].

Lemma 3.6.1. *The matrices H and H_+ in Algorithm 3.5.3 satisfy*

$$\frac{\det H_+}{\det H} = -\frac{s^T g}{s^T y}.$$

Proof This is shown in Exercise 8.9 by Nocedal and Wright [28]. □

Proposition 3.6.2. *Suppose zero is in the compact convex set C. We then deduce that the matrices H and H_+ in Algorithm 3.5.3 satisfy*

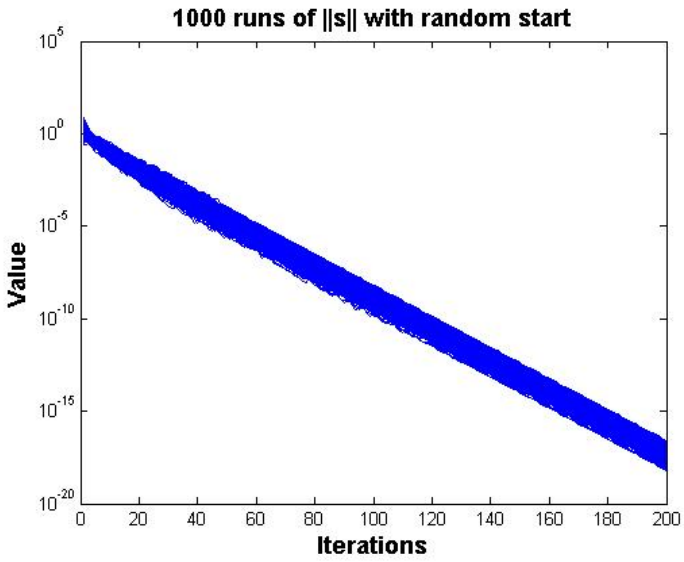
$$\det H_+ \leq \frac{\det H}{1 + \text{sym}(C)}.$$

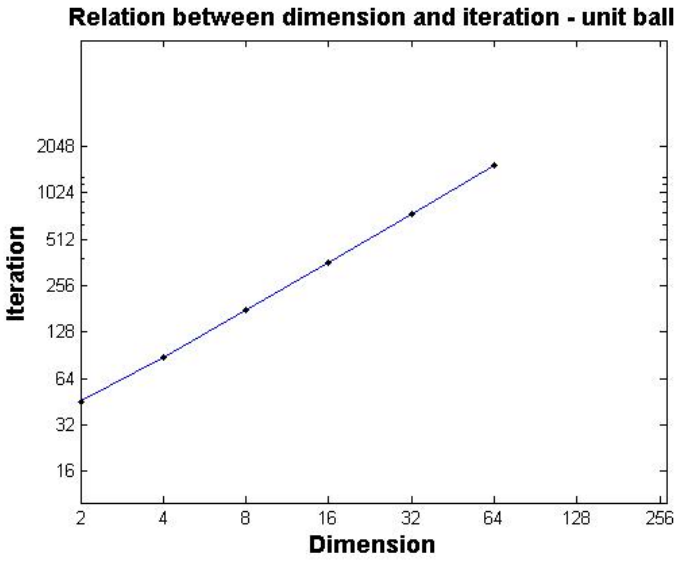
Proof

$$\begin{aligned}\frac{\det H}{\det H_+} = \frac{s^T y}{s^T g} &= \frac{s^T g - s^T g_+}{s^T g} \\ &= 1 + \frac{\max_C \langle \cdot, s\rangle}{-s^T g} \geq 1 + \frac{\langle -\text{sym}(C) g, s\rangle}{-s^T g} \\ &= 1 + \text{sym}(C).\end{aligned}$$

□

When C is a unit ball, f becomes simply the norm function. Through a number of numerical experiments, we arrive at the following conjecture.





Conjecture 3.6.3 (BFGS for the unit ball)**.** *Given any initial unit vector $g \in \mathbf{R}^n$ and matrix $H \in S^n_{++}$, if we repeatedly set*

$$s = -Hg, \;\; g_+ = \frac{s}{\|s\|}, \;\; y = g_+ - g, \;\; V = I - \frac{sy^T}{s^T y}, \;\; H_+ = VHV^T + \frac{ss^T}{s^T y},$$

and update $g = g_+$ and $H = H_+$, then the trial step s converges to zero.

This figure plots a thousand runs of $\|s\|$ against iterations with random ini-

tial points in dimension $n = 5$, suggesting a linear convergence rate. The next figure shows the number of iterations to make $\|s\|$ reach to 10^{-8} against different dimensions. As we see, the number grows linearly roughly like $n^{\frac{1}{\sqrt{2}}}$.

Define $E \succ F$ for matrices $E, F \in S^n$ to mean $E - F \in S^n_{++}$.

Theorem 3.6.4. *The matrices H and H_+ in Conjecture 3.6.3 satisfy*

$$\det H_+ \leq \frac{1}{2} \det H \ \ \textit{and} \ \ \lambda_{\max}(H_+) \leq \lambda_{\max}(H).$$

Proof The first inequality can be deduced from Proposition 3.6.2. Before we prove the second inequality, note that the BFGS update has two useful properties.

- Invariant under scaling: if we replace H by γH, then H_+ will be γH_+.
- Invariant under orthogonal transformation: if we replace H by $U^T H U$ for an orthogonal matrix U, and g by $U^T g$, then H_+ will be $U^T H_+ U$.

In this way, without loss of generality, we can assume

$$s = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ \ g = -\begin{bmatrix} \alpha \\ b \end{bmatrix},$$

for some scalar $\alpha \in (0, 1]$ and vector $b \in \mathbf{R}^{n-1}$ satisfying $\alpha^2 + \|b\|^2 = 1$. Therefore, H^{-1} can be represented by

$$H^{-1} = \begin{bmatrix} \alpha & b^T \\ b & E \end{bmatrix},$$

as $s = -Hg$. Moreover, $E \in S^{n-1}$ satisfies $E \succ \frac{bb^T}{\alpha}$, since H^{-1} is a positive definite matrix.

In fact, H^{-1} serves as the approximate Hessian matrix. Then the corresponding BFGS update becomes

$$H_+^{-1} = H^{-1} + \frac{yy^T}{s^T y} + \frac{gg^T}{s^T g},$$

so specifically

$$H_+^{-1} = \begin{bmatrix} 1+\alpha & b^T \\ b & E - \frac{bb^T}{\alpha(1+\alpha)} \end{bmatrix}.$$

For any $\lambda \in \{x \in \mathbf{R}_+ : x < \lambda_{\min}(H^{-1})\} = \{x \in \mathbf{R}_+ : x < (\lambda_{\max}(H))^{-1}\}$, we know

$$H^{-1} - \lambda I = \begin{bmatrix} \alpha - \lambda & b^T \\ b & E - \lambda I \end{bmatrix} \succ 0,$$

which is equivalent to

$$\alpha - \lambda > 0 \text{ and } E - \lambda I \succ \frac{bb^T}{\alpha - \lambda},$$

by the Schur complement.

Therefore, we can deduce $1 + \alpha - \lambda > 0$ and

$$\begin{aligned} E - \frac{bb^T}{\alpha(1+\alpha)} - \lambda I &\succ \frac{bb^T}{\alpha - \lambda} - \frac{bb^T}{\alpha(1+\alpha)} \\ &= (\frac{\alpha^2 + \lambda}{(\alpha - \lambda)\alpha(1+\alpha)} bb^T \\ &\succ 0 \end{aligned}$$

meaning

$$H_+^{-1} - \lambda I = \begin{bmatrix} 1 + \alpha - \lambda & b^T \\ b & E - \frac{bb^T}{\alpha(1+\alpha)} - \lambda I \end{bmatrix}$$

is also positive definite.

Hence, $\lambda < \lambda_{min}(H_+^{-1}) = (\lambda_{\max}(H_+))^{-1}$. □

3.7 Cholesky factors and line segments

Recall the update for Shor's R-algorithm is $s = -V^T V g$ and that for BFGS is $s = -Hg$. Intuitively, we should treat $H \approx V^T V$ in order to make a fair comparison. Since the inverse Hessian approximation is always symmetric positive definite, then it is useful to update the Cholesky factor of H directly [8, 33]. Denote $H = T^T T$ and $H_+ = T_+^T T_+$. Then, after some calculation,

$$T_+ = T(I - qs^T), \text{ where } q = \frac{y}{s^T y} + \frac{g}{\sqrt{-s^T g s^T y}}.$$

After a change of variable $h = Tg$, $p = Tg_+$ and conv $Q = TC$, Algorithm 3.5.3 becomes the following.

Algorithm 3.7.1 (Cholesky BFGS for $0 \in \text{conv } Q$)**.**

Choose $h \in Q$;
for $k = 0, 1, 2, \ldots$ **do**
  Find a minimizer p of $\langle \cdot, h \rangle$ over Q;
  if $p^T h > 0$ **then**
    terminate with "$0 \notin \text{conv } Q$";
  end if
  $e = h - p$; $\beta = h^T e$; $W = I - \frac{eh^T}{\beta} + \frac{hh^T}{\|h\|\sqrt{\beta}}$; $Q = WQ$; $h = Wp$;
end for

Returning to the polyhedral example of minimizing $f(x) = \max_{a_i \in Q} a_i^T x$ in Section 3.2, consider $Q = \text{conv} a_i \in \mathbf{R}^n$ indexed by a finite set I. The above algorithm is equivalent to:

Algorithm 3.7.2 (Cholesky BFGS for $0 \in \text{conv}\{a_i : i \in I\}$)**.**

Choose $i \in I$;

for $k = 0, 1, 2, \ldots$ **do**

Find $j \in I$ minimizing $a_i^T a_j$;

if $a_i^T a_j > 0$ **then**

terminate with "0 lies outside the convex hull";

end if

$e = a_i - a_j$; $\beta = a_i^T e$;

for each $r \in I$ **do**

$a_r = a_r - (a_i^T a_r)(\frac{e}{\beta} - \frac{a_i}{\|a_i\|\sqrt{\beta}})$;

end for

$i = j$;

end for

To illustrate, consider the special case where $|I| = 2$ with two distinct vectors a_1 and a_2 in $\mathbf{R}^n$. We can derive a simple algorithm as follows.

Algorithm 3.7.3 (Cholesky BFGS for $0 \in [c, d]$)**.**

for $k = 0, 1, 2, \ldots$ **do**

if $c^T d > 0$ **then**

terminate with "$0 \notin [c, d]$";

end if

$e = c - d$; $\beta = c^T e$;

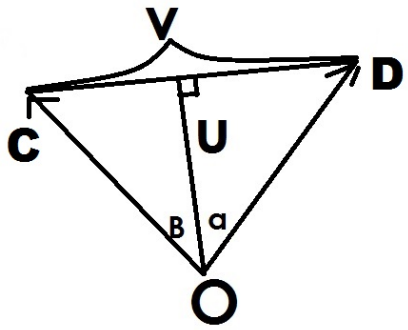
$d_+ = d - (c^T d)(\frac{e}{\beta} - \frac{c}{\|c\|\sqrt{\beta}})$; $c_+ = c - (c^T c)(\frac{e}{\beta} - \frac{c}{\|c\|\sqrt{\beta}})$;

$c = d_+$; $d = c_+$;

end for

Before showing the analysis of convergence, we introduce a measure

$$\gamma[c,d] \;=\; \frac{\sqrt{\|c\|^2\|d\|^2 - (c^T d)^2}}{\|c-d\|^2}.$$



To understand this measure, consider u and v in the graph

$$\begin{aligned} u^T v &= 2 \times \text{area of triangle} = |c||d| \sin\theta \\ &= |c||d| \sqrt{1-\cos^2\theta} = |c||d| \sqrt{1 - (\frac{c^T d}{\|c\|\|d\|})^2} \\ &= \sqrt{\|c\|^2\|d\|^2 - (c^T d)^2}. \end{aligned}$$

Therefore,

$$\gamma[c,d] = \frac{\sqrt{\|c\|^2\|d\|^2 - (c^T d)^2}}{\|c-d\|^2} = \frac{u}{v} = \frac{1}{\tan\alpha + \tan\beta}$$

Note that this measure is invariant under scaling and orthogonal transformations:

$$\begin{aligned} \gamma(\alpha[c,d]) &= \gamma[c,d] \text{ for any nonzero scalar } \alpha \\ \gamma(U[c,d]) &= \gamma[c,d] \text{ for any } n\text{-by-}n \text{ orthogonal matrix } U. \end{aligned}$$

Under these two properties, if we choose a basis carefully, then without lose of generality we can set

$$a_1 = c = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{and} \quad a_2 = d = \begin{bmatrix} -p \\ q \end{bmatrix} \quad \text{with } q \geq 0,$$

We only consider two-dimensional space, because vectors in Algorithm 3.7.3 only evolve in the two-dimensional space spanned by c and d. In this case, we can compute

$$\gamma[c,d] = \frac{q}{(1+p)^2+q^2}.$$

Lemma 3.7.4 (Improved conditioning). *If $c^T d \le 0$, then*

$$\gamma[c_+,d_+] \ge \gamma[c,d] + (\gamma[c,d])^3.$$

Proof At first, we can deduce the following from Algorithm 3.7.3:

$$c_+ = \begin{bmatrix} \frac{-1}{\sqrt{1+p}} \\ \frac{q}{1+p} \end{bmatrix} \quad \text{and} \quad d_+ = \begin{bmatrix} \frac{p}{\sqrt{1+p}} \\ \frac{q}{1+p} \end{bmatrix},$$

so

$$\gamma[c_+,d_+] = \frac{q}{(1+p)^{3/2}}.$$

Therefore,

$$\begin{aligned} \frac{\gamma[c_+,d_+]}{\gamma[c,d]} = \frac{\frac{q}{(1+p)^{3/2}}}{\frac{q}{(1+p)^2+q^2}} &= \frac{(1+p)^2+q^2}{(1+p)^{3/2}} \\ &= (1+p)^{\frac{1}{2}} + \frac{q^2}{(1+p)^{\frac{3}{2}}} \\ &\ge 1 + (\frac{q}{(1+p)^{\frac{3}{4}}})^2 \\ &\ge 1 + (\gamma[c,d])^2. \end{aligned}$$

□

Lemma 3.7.5.

$$\gamma_k \ge \gamma_0 + k\gamma_0^3 \quad \textit{for all} \quad k \in \mathbf{R}_+$$

Proof Prove by induction. When $k = 0$, this base case is trivial. Suppose this inequality holds for $k = n$, which is $\gamma_n \geq \gamma_0 + n\gamma_0^3$. When $k = n + 1$,

$$\gamma_{n+1} \geq \gamma_n + \gamma_n^3 \geq \gamma_0 + n\gamma_0^3 + (\gamma_0 + n\gamma_0^3)^3 \geq \gamma_0 + (n+1)\gamma_0^3.$$

□

Theorem 3.7.6. *For any distinct nonzero vectors $c, d \in \mathbf{R}^n$, if the line segment $[c, d]$ does not contain zero, then, after a number of iterations not exceeding*

$$\frac{\|c - d\|^6}{2[\|c\|^2\|d\|^2 - (c^T d)^2]^{\frac{3}{2}}}$$

Algorithm 3.7.3 terminates correctly.

Proof Suppose the algorithm does not terminate. Then we have:

- $\gamma > 0$. Check termination condition.
- $\gamma < \frac{1}{2}$. Check $(1 + p)^2 + (q - 1)^2 \geq 1$ for γ_0.

By previous lemma, after $\frac{1}{2\gamma_0^3}$ iterations, we have $\gamma \geq \gamma_0 + \frac{1}{2} > \frac{1}{2}$. The theorem is proved by contradiction. □

Note that we can shrink this bound to $\frac{\|c-d\|^4}{\|c\|^2\|d\|^2-(c^T d)^2}$ with a more complex argument [18].

CHAPTER 4

A COMPARISON OF BFGS AND SR1 FOR NONSMOOTH OPTIMIZATION

4.1 Introduction

Variable metric methods are well-known ways to solve numerical optimization problems. When it comes to nonsmooth optimization, the BFGS method seems to solve such problems effectively, as we indicated in previous chapters. A natural idea is to test the BFGS method along with alternative variable metric methods, in particular the symmetric rank one (SR1) method and Shor's R-algorithm on different test functions.

Tests show that the BFGS method is the best among all three methods in general, though SR1 is better than BFGS in some cases. Since a number of comparisons between BFGS and Shor's R-algorithm have been done in a previous chapter, this chapter mainly focuses on an exploration of the SR1 method for nonsmooth optimization, together with a comparison between this method and the BFGS algorithm.

We introduce the SR1 algorithm in the next section. Interestingly, a concrete example in the third section shows an SR1 trust region method, a popular method for smooth optimization, is not helpful for nonsmooth cases. In addition, we construct a closed-form example for the BFGS and SR1 methods to discuss the behavior under an exact line search setting. Moreover, we explore various modifications to the standard SR1 algorithm, followed by systematic numerical experiments. Finally, we draw a conclusion.

4.2 The SR1 Algorithm

The symmetric rank one algorithm is a popular Quasi-Newton method. Unlike BFGS, SR1 update has the general form

$$B_+ = B + \sigma v v^T$$

where B represents the approximate Hessian and v is a vector (In previous chapters, we update the the approximate inverse Hessian $H = B^{-1}$). Specifically, if y and s are difference between two successive gradients and points respectively, the new approximate Hessian B_+ has the following form:

$$B_+ = B + \frac{(y - Bs)(y - Bs)^T}{(y - Bs)^T s}.$$

In fact, the SR1 update is the only rank one update satisfying the secant condition where $y = B_+ s$. In contrast, the BFGS update is not the only rank two update satisfying this condition. For example, the DFP (Davidon, Fletcher, and Powell) update

$$B_+ = (I - \frac{ys^T}{y^T s})B(I - \frac{sy^T s}{y^T s}) + \frac{yy^T}{y^T s},$$

discovered even earlier than the BFGS update, also meets all requirements. Despite the simplicity of the SR1 update, it often generates a very good Hessian approximation, even better than BFGS updates [28].

There is an assumption for this update, namely $(y - Bs)^T s \neq 0$. If $y - Bs = 0$, then we set $B_+ = B$. On the other hand, if $y \neq Bs$ and $(y - Bs)^T s = 0$, then the algorithm fails. To prevent this, a simple safeguard must be applied in practice. Specifically, for small γ, typically around 10^{-8}, we only make the SR1 update if

$$|s^T(y - Bs)| \geq \gamma \|s\| \|y - Bs\|$$

holds. Otherwise, we skip the update and set $B_+ = B$ [28].

SR1 updating does not maintain the approximate Hessian B_+ to be positive definite, even in convex optimization with a positive definite B. In general, if the current point is far from the minimizer of a nonconvex problem, then indefinite Hessian approximation may be desirable, because it reflects the indefiniteness of the true Hessian. The only issue is that the search direction $-B^{-1}\nabla f(x)$ may not be a descent direction, so the assumption of the Armijo-Wolfe line search fails.

There are two common ways of using the approximate Hessian generated by SR1.

- Trust region: We discuss the SR1 trust region method in the next section. The advantage is that a trust region method does not require the Hessian approximation to be positive definite. As a matter of fact, it is also widely used in smooth constrained optimization in practice. However, this idea fails on nonsmooth optimization, as we shall see.
- Line search with Hessian modification: We discuss variations of line search algorithms, together with three different methods for generating descent search directions. We are interested in this approach, because it leads to a fair comparison with the standard BFGS algorithm and turns out ot be a reasonable solution in nonsmooth situations.

4.3 The SR1 Trust Region method

If we consider the approximate Hessian in the SR1 method as local metric information, then the trust region method is a classic technique to find a better local

point. This algorithm can be considered as:

1. Find a "good" point using the approximate Hessian matrix in a trust region defined by the current point and radius.
2. Calculate the actual and predicted reduction by the current model at the new point respectively.
3. If the ratio of the actual over the predicted is greater than a prefixed threshold, then move to the new point and increase the radius. Otherwise, stay at the current point and decrease the radius.
4. No matter whether the current point moves or not, the SR1 update is always performed in order to learn curvature information better.

The numerical algorithms we discussed always compress all the previous information into the model at the most recent point. The purpose of a trust region method is to find the next point not too far from the current point, intuitively, so that we still trust the model. Furthermore, a successful step 3 leads to a larger step by expending the radius, and a failure results in a smaller step by shrinking the radius.

The SR1 trust region algorithm is $(n+1)$ - step superlinearly convergent even without the assumption of positive definiteness [34]. It is natural to ask ourselves whether this method can be applied to nonsmooth optimization, since the BFGS algorithm is effective in that setting. However, the behavior of a simplified SR1 trust region method, which we present below, does not converge to optimal solution even for as simple a nonsmooth function as $f(x_1, x_2) = k|x_1| + x_2^2$ where $k > 0$. Numerical experiments support this observation from the literature [24]. In short, the reason of failure is that when current iterate is close to the

nonsmooth edge ($x_1 = 0$), then the new step is mainly towards the nonsmooth edge instead of the direction $x_2 = 0$. A specific counterexample follows.

4.3.1 Simplified Algorithm

In this section, we introduce a simplified version of the SR1 trust region method. The key subproblem of trust region methods is to solve

$$\min\ m(p) = f + \nabla f^T p + \frac{1}{2} p^T B p \quad \text{s.t.} \quad \|p\| \leq \Delta, \tag{3.1}$$

where the function value $f(x + p)$ is approximated by $m(p)$ at current point x. A solution p of (3.1) satisfies the formula

$$(B + \lambda I)p^* = -\nabla f \tag{3.2}$$

for some $\lambda \geq 0$.

The following theorem provides us an insight of the Levenberg-Marquardt method, the most important type of restricted step method first suggested by Levenberg and Marquardt in the context of nonlinear least squares problem [13].

Theorem 4.3.3. *[13, Thm 5.2.1] p^* is a global solution of (3.1) if and only if there exists $\lambda \geq 0$ such that (3.2) holds, $\lambda(\Delta - \|p^*\|) = 0$, and $B + \lambda I$ is positive semi-definite.*

In addition, if $B + \lambda I$ is positive definite, then p^* is the unique solution of (3.1).

Theorem 4.3.3 is concerned with global optimal solution, and this shows the significance of formula 3.2. Since the increase of λ cause $\|p\|$ to decrease, and vice versa, then it is possible to construct a trust region-like algorithm except that changes to Δ are replaced by changes to λ. Moreover, it also simplifies

the standard trust region method due to the computational difficulty of solving (3.1).

Algorithm 4.3.4 (Simplified SR1 trust region)**.**

Given x_0, B, $\rho > 0$, $r = 10^{-8}$

for $k = 0, 1, 2, \ldots$ **do**

$s = -(B + \rho I)^{-1}\nabla f(x)$; $x_+ = x + s$

$y = \nabla f(x_+) - \nabla f(x)$; $v = y - Bs$

if $v \neq 0$ and $|s^T v| \geq r\|s\| \cdot \|v\|$ **then**

$B_+ = B + \frac{vv^T}{v^T s}$

end if

$ared = f(x) - f(x_+)$, and $pred = -[\nabla f(x)^T s + \frac{1}{2}s^T(B + \rho I)s]$

if $\frac{ared}{pred} \geq \frac{1}{2}$ **then**

$x = x_+$ and $\rho_+ = \frac{\rho}{2}$

else

$\rho_+ = 2\rho$

end if

$B = B_+$; $\rho = \rho_+$

end for

Along iterations, the point x may move to a new point or stay at its current position alternatively. The first question we should ask ourselves is whether this algorithm may stick at a point with $\nabla f(x) \neq 0$ forever. The answer is no, because when ρ is large enough, the algorithm is similar to the steepest descent method with an arbitrarily small step size. Since $-\nabla f$ is a descent direction, meaning giving a negative directional derivative, then it will move to a new point eventually.

4.3.2 A counter example

Before we introduce the counter example, there are two properties, which are useful for proving the failure of convergence.

Proposition 4.3.5. *Suppose Algorithm 4.3.4 is applied to the function* $f(x) = k|x^{(1)}| + (x^{(2)})^2$. *Given a current point* $x = \begin{pmatrix} x^{(1)} \\ x^{(2)} \end{pmatrix}$ *and a diagonal* $B = \begin{pmatrix} a & 0 \\ 0 & 2 \end{pmatrix}$ *where* $a \geq 0$, $\rho > 0$, *if we successfully update* x *to* x_+ *meaning* $f(x_+) < f(x)$, *and also if* $x^{(1)}$ *and* $x_+^{(1)}$ *have the same sign (neither being 0), then* $B_+ = \begin{pmatrix} 0 & 0 \\ 0 & 2 \end{pmatrix}$. *On the other hand, if* $x^{(1)}$ *and* $x_+^{(1)}$ *have different signs (neither being 0), then* $B_+ = \begin{pmatrix} 2(a+\rho) & 0 \\ 0 & 2 \end{pmatrix}$.

Proof Note that the only nonsmooth edge is $x^{(1)} = 0$ in this problem. Thus, it divides the domain into two smooth regions $\{x : x^{(1)} > 0\}$ and $\{x : x^{(1)} < 0\}$. $x^{(1)}$ and $x_+^{(1)}$ having same sign (not 0) is equivalent to both x and x_+ are from the same smooth region.

Without loss of generality, if $x^{(1)}$ and $x_+^{(1)}$ have the same sign (neither being 0), we assume $x^{(1)} > 0$ and $x_+^{(1)} > 0$.

$$s = -(B + \rho I)^{-1} \nabla f(x) = -\begin{pmatrix} \frac{1}{a+\rho} & 0 \\ 0 & \frac{1}{2+\rho} \end{pmatrix} \begin{pmatrix} k \\ 2x^{(2)} \end{pmatrix},$$

$$x_+ = x + s = \begin{pmatrix} x^{(1)} - \frac{k}{a+\rho} \\ x^{(2)} - \frac{2}{2+\rho} x^{(2)} \end{pmatrix}.$$

As $x_+^{(1)} > 0$, then $x^{(1)} > \frac{k}{a+\rho}$.

$$y = \nabla f(x_+) - \nabla f(x) = \begin{pmatrix} k \\ 2x^{(2)} - \frac{4}{2+\rho}x^{(2)} \end{pmatrix} - \begin{pmatrix} k \\ 2x^{(2)} \end{pmatrix} = \begin{pmatrix} 0 \\ -\frac{4}{2+\rho}x^{(2)} \end{pmatrix},$$

$$v = y - Bs = \begin{pmatrix} 0 \\ -\frac{4}{2+\rho}x^{(2)} \end{pmatrix} + \begin{pmatrix} a & 0 \\ 0 & 2 \end{pmatrix}\begin{pmatrix} \frac{k}{a+\rho} \\ \frac{2}{2+\rho}x^{(2)} \end{pmatrix} = \begin{pmatrix} \frac{ka}{a+\rho} \\ 0 \end{pmatrix},$$

$$B_+ = B + \frac{vv^T}{v^T s} = \begin{pmatrix} a & 0 \\ 0 & 2 \end{pmatrix} + \begin{pmatrix} -\frac{ka}{a+\rho}\frac{a+\rho}{k} & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 2 \end{pmatrix}.$$

On the other hand, if $x^{(1)}$ and $x_+^{(1)}$ have different signs (neither being 0), we can set $x^{(1)} > 0$ and $x_+^{(1)} < 0$, without loss of generality.

$$s = -(B + \rho I)^{-1}\nabla f(x) = -\begin{pmatrix} \frac{1}{a+\rho} & 0 \\ 0 & \frac{1}{2+\rho} \end{pmatrix}\begin{pmatrix} k \\ 2x^{(2)} \end{pmatrix},$$

$$x_+ = x + s = \begin{pmatrix} x^{(1)} - \frac{k}{a+\rho} \\ x^{(2)} - \frac{2}{2+\rho}x^{(2)} \end{pmatrix}.$$

As $x_+^{(1)} < 0$, then $x^{(1)} < \frac{k}{a+\rho}$.

$$y = \nabla f(x_+) - \nabla f(x) = \begin{pmatrix} -k \\ 2x^{(2)} - \frac{4}{2+\rho}x^{(2)} \end{pmatrix} - \begin{pmatrix} k \\ 2x^{(2)} \end{pmatrix} = \begin{pmatrix} -2k \\ -\frac{4}{2+\rho}x^{(2)} \end{pmatrix},$$

$$v = y - Bs = \begin{pmatrix} -2k \\ -\frac{4}{2+\rho}x^{(2)} \end{pmatrix} + \begin{pmatrix} a & 0 \\ 0 & 2 \end{pmatrix}\begin{pmatrix} \frac{k}{a+\rho} \\ \frac{2}{2+\rho}x^{(2)} \end{pmatrix} = \begin{pmatrix} -2k + \frac{ka}{a+\rho} \\ 0 \end{pmatrix},$$

$$B_+ = B + \frac{vv^T}{v^T s} = \begin{pmatrix} a & 0 \\ 0 & 2 \end{pmatrix} + \begin{pmatrix} (2k - \frac{ka}{a+\rho})\frac{a+\rho}{k} & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 2(a+\rho) & 0 \\ 0 & 2 \end{pmatrix}.$$

□

Theorem 4.3.6. *Consider the function* $f(x) = 24|x^{(1)}| + (x^{(2)})^2$, *initial point* $x_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $\rho_0 = 1$, *and initial approximate Hessian* $B_0 = \begin{pmatrix} 0 & 0 \\ 0 & 2 \end{pmatrix}$. *Let* $x_i = \begin{pmatrix} x_i^{(1)} \\ x_i^{(2)} \end{pmatrix}$ *be the sequence of distinct iterates after successful updates. Then Algorithm 4.3.4 results in* $x_i^{(1)} = \frac{1}{\rho_i} = \frac{1}{4^{i-1}}$, $x_i^{(2)} \geq 0.8 - \sum_{j=1}^{i-1} \frac{1}{4^j} \geq \frac{7}{15}$, $\rho_i = 4^{i-1}$ *and* $B_i = B_0$ *for all* i.

Proof : We prove by induction. It is easy to check the base case.

Assume $x_k = (x_k^{(1)}, x_k^{(2)})$ where $x_k^{(1)} = \frac{1}{\rho_k} = \frac{1}{4^{k-1}}$, $B_k = B_0$ for $k = n \in \mathcal{N}$, and check the case $k = n + 1$.

In general, we do several iterations before a successful update. This problem requires four such iterations in total. Moreover, let $\rho_n^{[i]}$ and $B_n^{[i]}$ be the value of ρ and approximate Hessian B at the ith iteration after n successful updates on point x.

- Iteration 1:

$$s = -(B_n + \rho_n I)^{-1} \nabla f(x) = -\begin{pmatrix} \frac{1}{4^{n-1}} & 0 \\ 0 & \frac{1}{2+4^{n-1}} \end{pmatrix} \begin{pmatrix} 24 \\ 2x_n^{(2)} \end{pmatrix},$$

$$x_+ = x_n + s = \begin{pmatrix} \frac{1}{4^{n-1}} - \frac{24}{4^{n-1}} \\ x_n^{(2)} - \frac{2}{2+4^{n-1}} x_n^{(2)} \end{pmatrix}.$$

  Thus, $f(x_+) = \frac{24 \cdot 23}{4^{n-1}} + (\frac{4^{n-1}}{2+4^{n-1}} x_n^{(2)})^2$, and $f(x_n) = \frac{24}{4^{n-1}} + (x_n^{(2)})^2$.

  We can check that since $x_n^{(2)} \leq 1$ by assumption, then $f(x_+) > f(x_n)$, so x_n does not move to x_+.

By Proposition 4.3.5, $B_n^{[1]} = B_+ = \begin{pmatrix} 2\rho_n & 0 \\ 0 & 2 \end{pmatrix} = \begin{pmatrix} 2 \cdot 4^{n-1} & 0 \\ 0 & 2 \end{pmatrix}$ as $x_+^{(1)} < 0$, and $\rho_n^{[1]} = 2\rho_n = 2 \cdot 4^{n-1}$.

- Iteration 2:

$$s = -(B_n^{[1]} + \rho_n^{[1]} I)^{-1} \nabla f(x) = -\begin{pmatrix} \frac{1}{4 \cdot 4^{n-1}} & 0 \\ 0 & \frac{1}{2+2 \cdot 4^{n-1}} \end{pmatrix} \begin{pmatrix} 24 \\ 2x_n^{(2)} \end{pmatrix},$$

$$x_+ = x_n + s = \begin{pmatrix} \frac{1}{4^{n-1}} - \frac{6}{4^{n-1}} \\ x_n^{(2)} - \frac{2}{2+2 \cdot 4^{n-1}} x_n^{(2)} \end{pmatrix}.$$

Thus, $f(x_+) = \frac{24 \cdot 5}{4^{n-1}} + (\frac{2 \cdot 4^{n-1}}{2+2 \cdot 4^{n-1}} x_n^{(2)})^2$, and $f(x_n) = \frac{24}{4^{n-1}} + (x_n^{(2)})^2$.

We can check that since $x_n^{(2)} \leq 1$ by assumption, then $f(x_+) > f(x_n)$, so x_n does not move to x_+.

By Proposition 4.3.5, $B_n^{[2]} = B_+ = \begin{pmatrix} 2(2\rho_n + \rho_n^{[1]}) & 0 \\ 0 & 2 \end{pmatrix} = \begin{pmatrix} 8 \cdot 4^{n-1} & 0 \\ 0 & 2 \end{pmatrix}$ as $x_+^{(1)} < 0$, and $\rho_n^{(2)} = 2\rho_n^{(1)} = 4 \cdot 4^{n-1}$.

- Iteration 3:

$$s = -(B_n^{[2]} + \rho_n^{[2]} I)^{-1} \nabla f(x) = -\begin{pmatrix} \frac{1}{12 \cdot 4^{n-1}} & 0 \\ 0 & \frac{1}{2+4 \cdot 4^{n-1}} \end{pmatrix} \begin{pmatrix} 24 \\ 2x_n^{(2)} \end{pmatrix},$$

$$x_+ = x_n + s = \begin{pmatrix} \frac{1}{4^{n-1}} - \frac{2}{4^{n-1}} \\ x_n^{(2)} - \frac{2}{2+4 \cdot 4^{n-1}} x_n^{(2)} \end{pmatrix}.$$

Thus, $f(x_+) = \frac{24}{4^{n-1}} + (\frac{4 \cdot 4^{n-1}}{2+4 \cdot 4^{n-1}} x_n^{(2)})^2$, and $f(x_n) = \frac{24}{4^{n-1}} + (x_n^{(2)})^2$. We now find

$ared = f(x_n) - f(x_+) = (\frac{2}{2+4 \cdot 4^{n-1}} x_n^{(2)})^2$,

$pred = -[\nabla f(x_n)^T s + \frac{1}{2} s^T (B_n^{[2]} + \rho_n^{[2]} I) s] = \frac{24}{4^{n-1}} + \frac{2}{2+4 \cdot 4^{n-1}} (x_n^{(2)})^2$.

We can check that since $x_n^{(2)} \leq 1$ by assumption, then $\frac{ared}{pred} < \frac{1}{2}$, so x_n does not move to x_+.

By Proposition 4.3.5, $B_n^{[3]} = B_+ = \begin{pmatrix} 2(8 \cdot 4^{n-1} + \rho_n^{[2]}) & 0 \\ 0 & 2 \end{pmatrix} = \begin{pmatrix} 24 \cdot 4^{n-1} & 0 \\ 0 & 2 \end{pmatrix}$ as $x_+^{(1)} < 0$, and $\rho_n^{[3]} = 2\rho_n^{[2]} = 8 \cdot 4^{n-1}$.

- Iteration 4:

$$s = -(B_n^{[3]} + \rho_n^{[3]} I)^{-1} \nabla f(x) = -\begin{pmatrix} \frac{1}{32 \cdot 4^{n-1}} & 0 \\ 0 & \frac{1}{2+8 \cdot 4^{n-1}} \end{pmatrix} \begin{pmatrix} 24 \\ 2x_n^{(2)} \end{pmatrix},$$

$$x_+ = x_n + s = \begin{pmatrix} \frac{1}{4^{n-1}} - \frac{3}{4 \cdot 4^{n-1}} \\ x_n^{(2)} - \frac{2}{2+8 \cdot 4^{n-1}} x_n^{(2)} \end{pmatrix} = \begin{pmatrix} \frac{1}{4 \cdot 4^{n-1}} \\ x_n^{(2)} - \frac{2}{2+8 \cdot 4^{n-1}} x_n^{(2)} \end{pmatrix}.$$

Thus, $f(x_+) = \frac{24}{4 \cdot 4^{n-1}} + (\frac{4 \cdot 4^{n-1}}{2+4 \cdot 4^{n-1}} x_n^{(2)})^2$, and $f(x_n) = \frac{24}{4^{n-1}} + (x_n^{(2)})^2$. We now find

$ared = f(x_n) - f(x_+) = \frac{24 \cdot 3}{4 \cdot 4^{n-1}} + (\frac{4}{2+4 \cdot 4^{n-1}} x_n^{(2)})^2$,

$pred = -[\nabla f(x_n)^T s + \frac{1}{2} s^T (B_n^{[3]} + \rho_n^{[3]} I) s] = \frac{24}{4^{n-1}} + \frac{2}{2+4 \cdot 4^{n-1}} (x_n^{(2)})^2$.

We can check that since $\frac{2}{3} \leq x_n^{(2)} \leq 1$ by assumption, then $\frac{ared}{pred} \geq \frac{1}{2}$, so x_n does move to x_+. In all,

$$x_{n+1} = x_+ = \begin{pmatrix} \frac{1}{4 \cdot 4^{n-1}} \\ x_n^{(2)} - \frac{2}{2+8 \cdot 4^{n-1}} x_n^{(2)} \end{pmatrix} = \begin{pmatrix} \frac{1}{4^n} \\ x_n^{(2)} - \frac{2}{2+8 \cdot 4^{n-1}} x_n^{(2)} \end{pmatrix}.$$

By Proposition 4.3.5, $B_{n+1} = B_+ = \begin{pmatrix} 0 & 0 \\ 0 & 2 \end{pmatrix}$, and $\rho_{n+1} = \frac{\rho_n^{[3]}}{2} = 4 \cdot 4^{n-1} = 4^n$.

Moreover, $x_{n+1}^{(2)} = x_n^{(2)} - \frac{2}{2+8 \cdot 4^{n-1}} x_n^{(2)} \geq x_n^{(2)} - \frac{2}{2+8 \cdot 4^{n-1}} \geq x_n^{(2)} - \frac{2}{8 \cdot 4^{n-1}} = x_n^{(2)} - \frac{1}{4^n}$.

After we check that $x_1^{(2)} = 0.8$, then we get $x_n^{(2)} \geq 0.8 - \sum_{j=1}^{n-1} \frac{1}{4^j}$ if we iterate the above argument n times. Notice that $\sum_{j=1}^{\infty} \frac{1}{4^j} = \frac{1}{3}$ and $0.8 - \frac{1}{3} = \frac{7}{15}$. In summary, the above property follows by induction. □

This example is also easy to verify by numerical experiments. Indeed, ρ_i, B_i, and $x_i^{(1)}$ are exactly what the calculations indicate. In addition, when $x^{(1)} \approx 10^{-15}$, the corresponding $x^{(2)} \approx 0.7375$ unsurprisingly, because the lower bound $\frac{7}{15}$ is not tight.

To sum up, this section suggests that the SR1 trust region method does not work in nonsmooth optimization by providing a counter example. Notice that trust region method combines searching for a direction and choosing a step size simultaneously. When the current point is close to the nonsmooth line ($x_1 = 0$), then ρ has to be huge in order to make a move locally; however, such new step is mainly towards the nonsmooth edge and achieves little improvement in the direction of $x_2 = 0$.

This suggests why the line search seems to be important in nonsmooth optimization. Intuitively, the Wolfe conditions in the line search make the new point satisfy sufficient decrease but be not too close to the current point, in order to learn more gradient information. In the next sections, we explore the power of Quasi-Newton methods with line search.

4.4 Exact Line Search for Quasi-Newton Method

The standard SR1 algorithm with line search is stated as follows.

Algorithm 4.4.1 (SR1 with line search)**.**

Given x_0, H_0, $r = 10^{-8}$

for $k = 0, 1, 2, \ldots$ **do**

$d = -H\nabla f(x)$

$x_+ = x + \hat{t}d$ for some $\hat{t}$

$y = \nabla f(x_+) - \nabla f(x)$ and $v = s - Hy$ where $s = \hat{t}d = x_+ - x$

if $v \neq 0$ and $|v^T y| \geq r\|y\| \cdot \|v\|$ **then**

$H_+ = H + \frac{vv^T}{v^T y}$;

end if

$H = H_+$; $x = x_+$

end for

In this algorithm, the methodology of choosing $\hat{t}$ is not stated. There are two main ideas.

- Exact line search: $\hat{t} = \arg_{t>0} \min f(x + td)$, assuming d is a descent direction. This is often discussed in theory. However, such $\hat{t}$ may not be easy to calculate in practice.
- Inexact line search: this idea provides us a sufficiently good solution instead of the best point in a reasonable time. Therefore, we typically apply inexact line search in practice, though it complicates the analysis of convergence. In this chapter, whenever inexact line search is referred to, it means a Wolfe line search. More details will be addressed in later sections.

Since this section concentrates on exact line search, we should quote Dixon's theorem which states that, when applying exact line search, all methods in the Broyden family generate the same sequence of iterates x_k [11]. Since both the BFGS and SR1 methods belong to the Broyden family where $B_+ = (1 - \Phi)B_{BFGS} + \Phi B_{DFP}$ for some Φ, we cannot quite distinguish them in the case of exact line search.

To clarify, it is useful to construct a tractable illustration of the success of the BFGS and SR1 methods for non-smooth functions.

4.4.1 Example

Lewis and Zhang [24] provided a useful example where a sequence of BFGS iterates

$$(u_{2k}, v_{2k}) = (\rho^k, \frac{2\rho^{2k}}{5}), \quad (u_{2k+1}, v_{2k+1}) = (\frac{\rho^k}{2}, -\frac{2\rho^{2k+1}}{5})$$

on the function $f(u, v) = u^2 + |v|$ are iterates of the exact-line-search BFGS method with explicit Hessian formula

$$H_{2k}^{BFGS} = \frac{1}{6}\begin{pmatrix} 5 & 2\rho^k \\ 2\rho^k & 8\rho^{2k} \end{pmatrix}, \quad H_{2k+1}^{BFGS} = \frac{1}{6}\begin{pmatrix} 5 & -\rho^k \\ -\rho^k & 8\rho^{2k+1} \end{pmatrix}$$

given $\rho = \frac{1}{4}$, and $k = 1, 2, 3, ...$

By Dixon's theorem, we know that the exact-line-search SR1 method generates the same sequence of iterates. To understand both the theorem and SR1 updates better, we accurately provide details of the SR1 sequence, especially the Hessian updates.

Proposition 4.4.2. *The exact-line-search SR1 method will produce the same sequence of* (u_{2k}, v_{2k}) *and* (u_{2k+1}, v_{2k+1}) *with the same function and starting point. The explicit Hessian formula is*

$$H_k^{SR1} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \rho^k \end{pmatrix}.$$

Proof: We prove by induction. It is easy to check base case.

1. Suppose the current point is in form of (u_{2k}, v_{2k}).

$$d^{SR1} = -H_{2k}^{SR1}\nabla f_{2k} = -\begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \rho^{2k} \end{pmatrix}\begin{pmatrix} 2\rho^k \\ 1 \end{pmatrix} = -\begin{pmatrix} \rho^k \\ \rho^{2k} \end{pmatrix},$$

$$s^{BFGS} = (u_{2k+1}, v_{2k+1}) - (u_{2k}, v_{2k}) = -\begin{pmatrix} \frac{1}{2}\rho^k \\ \frac{1}{2}\rho^{2k} \end{pmatrix}.$$

As s^{BFGS} has same direction as d^{BFGS} and $s^{BFGS} = \frac{1}{2}d^{SR1}$, then if we start at points in form of (u_{2k}, v_{2k}), we will end at point (u_{2k+1}, v_{2k+1}). Therefore, $s^{BFGS} = s^{SR1}$.

$$y = \nabla f_{2k+1} - \nabla f_{2k} = \begin{pmatrix} \rho^k \\ -1 \end{pmatrix} - \begin{pmatrix} 2\rho^k \\ 1 \end{pmatrix} = \begin{pmatrix} -\rho^k \\ -2 \end{pmatrix},$$

$$v = s^{SR1} - H_{2k}^{SR1}y = -\begin{pmatrix} \frac{1}{2}\rho^k \\ \frac{1}{2}\rho^{2k} \end{pmatrix} + \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \rho^{2k} \end{pmatrix}\begin{pmatrix} \rho^k \\ 2 \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{3}{2}\rho^{2k} \end{pmatrix},$$

$$H_{2k+1}^{SR1} = H_{2k}^{SR1} + \frac{v^T v}{v^T y} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \rho^{2k} \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & -\frac{3}{4}\rho^{2k} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \rho^{2k+1} \end{pmatrix}, \text{ as we expect.}$$

2. Suppose the current point is in form of (u_{2k+1}, v_{2k+1}).

$$d^{SR1} = -H_{2k+1}^{SR1}\nabla f_{2k+1} = -\begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \rho^{2k+1} \end{pmatrix}\begin{pmatrix} \rho^k \\ -1 \end{pmatrix} = \begin{pmatrix} -\frac{1}{2}\rho^k \\ \frac{1}{4}\rho^{2k} \end{pmatrix},$$

$$s^{BFGS} = (u_{2k+2}, v_{2k+2}) - (u_{2k+1}, v_{2k+1}) = \begin{pmatrix} -\frac{1}{4}\rho^k \\ \frac{1}{8}\rho^{2k} \end{pmatrix}.$$

As s^{BFGS} has same direction as d^{BFGS} and $s^{BFGS} = \frac{1}{2}d^{SR1}$, then if we start at points in form of (u_{2k+1}, v_{2k+1}), we will end at point (u_{2k+2}, v_{2k+2}). Therefore, $s^{BFGS} = s^{SR1}$.

$$y = \nabla f_{2k+2} - \nabla f_{2k+1} = \begin{pmatrix} \frac{1}{2}\rho^k \\ 1 \end{pmatrix} - \begin{pmatrix} \rho^k \\ -1 \end{pmatrix} = \begin{pmatrix} -\frac{1}{2}\rho^k \\ 2 \end{pmatrix},$$

$$v = s^{SR1} - H_{2k+1}^{SR1}y = \begin{pmatrix} -\frac{1}{4}\rho^k \\ \frac{1}{8}\rho^{2k} \end{pmatrix} + \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{4}\rho^{2k} \end{pmatrix}\begin{pmatrix} \frac{1}{2}\rho^k \\ -2 \end{pmatrix} = \begin{pmatrix} 0 \\ -\frac{3}{8}\rho^{2k} \end{pmatrix},$$

$$H_{2k+2}^{SR1} = H_{2k+1}^{SR1} + \frac{v^T v}{v^T y} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{4}\rho^{2k} \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & -\frac{3}{16}\rho^{2k} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \rho^{2k+2} \end{pmatrix}, \text{ as we expect.}$$

□

In this example, we verify the correctness of Dixon's theorem, though the approximate Hessian matrices of these two methods vary a lot. In addition, the BFGS and SR1 methods always take step size to be $\frac{1}{4}$ and $\frac{1}{2}$ respectively for all iterations. Note that we may expect the Quasi-Newton method will eventually take unit step in smooth optimization. As we know, if the function f is three times continuously differentiable with some further assumptions, the BFGS algorithm converges to a minimizer superlinearly and eventually a unit step will be admissible, meaning it satisfies Wolfe conditions [28, Thm 3.5, Thm 8.6].

4.5 SR1 Method with Inexact Line Search

The Wolfe line search is a classic inexact line search method. Specifically, it requires $\hat{t}$ in Algorithm 4.4.1 to satisfy two conditions, assuming d is a descent direction:

$$f(x + \hat{t}d) \leq f(x) + c_1 \hat{t} \nabla f(x)^T d \tag{5.1}$$

$$\nabla f(x + \hat{t}d)^T d \geq c_2 \nabla f(x)^T d \tag{5.2}$$

with $0 < c_1 < c_2 < 1$.

Condition 5.1 ensures a sufficient decrease of function value. Here are three key aspects of it.

- If $c_1 \to 0_+$, then such x_{new} is acceptable as long as $f(x_{new}) < f(x)$.
- If $c_1 \to 1_-$, then the righthand side becomes first order Taylor approximation at x.
- In general, this inequality indicates the reduction of function value f should be a prefixed fraction of the directional derivative $\nabla f(x)^T d$ at least.

Condition 5.2 rules out an undesirable small step where x_+ is too close to x. If we denote $\phi(t) = f(x + td)$, then this condition becomes $\phi'(t) \geq c_2\phi'(0)$.

Assuming the function f is continuously differentiable and d is a descent direction, one can also show there always exists an interval of t satisfying both conditions, and one such $\hat{t}$ can always be found by standard line search techniques [28, Chapter 3].

Nevertheless, since SR1 method does not require the approximate inverse Hessian to be positive definite, it ruins the fundamental requirement of the line search, because the search direction d may not be a descent direction.

4.5.1 Three SR1 methods with modifications

We present three different modifications of Algorithm 4.4.1 in order to generate a reasonable descent direction to perform line search at each iteration, assuming the current approximate Hessian H is nonsingular.

SR1 with absolute Hessian

Consider a symmetric matrix H that may not be positive definite. $H = Q\Lambda Q^T$ by spectral decomposition where $\Lambda = \text{diag}(\lambda_i)$ and Q is an orthogonal matrix.

Define $|H| = Q|\Lambda|Q^T$ where $|\Lambda| = \text{diag}(|\lambda_i|)$. Clearly $|H|$ is positive definite. If we set $d = -|H|\nabla f(x)$, then a descent direction is generated, meaning the directional derivative along d is negative [28].

Bidirectional SR1 Method

If d is not a descent direction, then $-d$ is one due to the nonsingularity of H.

Note that $d = -H\nabla f(x) = -Q\Lambda Q^T\nabla f(x)$. Then $-d = -Q(-\Lambda)Q^T\nabla f(x)$. When d is not a descent direction, the bidirectional SR1 method flips signs of all eigenvalues; whereas SR1 with absolute Hessian flips just some of them.

SR1 Method with Negative Curvature

If d is not a descent direction, we know H is not positive definite. Thus, there exists an eigenvector v corresponding to a negative eigenvalue. Assuming $\nabla f(x)^T v \neq 0$, either v or $-v$ is a descent direction, along which we see the negative curvature [20, 30]. There are different methods to generate v, but it is unclear how to set the initial step size in a backtracking line search algorithm.

Recall from section 3.3, Newton's step provides rich information, including both the search direction, and the scale. Therefore, a standard Newton backtracking the algorithm always has initial step size to be one.

However, if choosing the unit eigenvector as a search direction, we have no idea about the approximate scale of acceptable step size $\hat{t}$. Literature suggests a heuristic algorithm by setting the initial scale to be the acceptable step size at the previous "active" iteration [30]. Here we define an "active" iteration to be an iteration where d is not a descent direction.

4.6 Numerical Experiments

Now we can compare BFGS, SR1 and Shor's R-algorithm with line search in a systematical way. To illustrate, we consider four different classes of functions, together with a very simple representative function in each class, listed below.

- Convex smooth class: quadratic function.
- Nonconvex smooth class: smooth Rosenbrock function.
- Convex nonsmooth class: norm function.
- Nonconvex nonsmooth class: max of quadratic functions.

Based on test results and analysis, we hope to provide some insights on the power of Quasi-Newton update in nonsmooth optimization particularly.

It turns out that all three SR1 methods have similar numerical performance and the version with the absolute Hessian seems slightly better. Therefore, when it comes to the SR1 method with line search, we only present results for SR1 with absolute Hessian modification, though all three different SR1 methods were tested.

4.6.1 Quadratic Functions

The simplest class of functions in convex smooth optimization are quadratic functions. The BFGS and SR1 methods are quite powerful for this type of functions. We recall a theorem about Broyden class under exact line search [28, Thm 8.4].

Theorem 4.6.1. *Suppose that $f : \mathbf{R}^n \to \mathbf{R}$ is a strongly convex quadratic function $f(x) = b^T x + \frac{1}{2} x^T A x$ where both A and initial Hessian B_0 are symmetric positive definite. Using exact line search and Broyden-class quasi-Newton updates:*

- *The iterates converge in at most n iterations.*
- *If n iterations are performed, we have $B_{n+1} = A$.*

In fact, the SR1 method will also terminate within n iterations for strongly convex quadratic functions regardless the choice of step length [28, Thm 8.1].

In our experiments, hundreds of such functions are generated with dimension $n = 5$, and the SR1 method terminates in 5 steps, as the theorem indicates. Since line search may involve multiple function evaluations, the total number of function evaluations are around 6 mostly for the SR1 method, if we choose the coefficients of Armijo-Wolfe to be 10^{-8} and 0.5 respectively from experiments. At the same time, the BFGS method, with the same initial conditions, terminates in total ranging from 7 to 13 function evaluations. Therefore, the SR1 method typically takes less evaluations than the BFGS method in this example.

However, Shor's R-algorithm with the same coefficient of line search typically takes a large number of iterations. For example, it takes more than 100 iterations to reach value 10^{-8} for the function $f(x) = x_1^2 + 2x_2^2 + 3x_3^2 + 4x_4^2$ if we

randomly generate a starting point from a unit ball with radius 5. Specially, we introduce Shor's R-algorithm with inexact line search.

Algorithm 4.6.2 (Shor with line search)**.**

Given x_0, V_0

for $k = 0, 1, 2, \ldots$ **do**

$d = -V^T V \nabla f(x)$

$x_+ = x + \hat{t}d$ for some $\hat{t}$

$e = V(\nabla f(x_+) - \nabla f(x))$

$W = I - \frac{2}{3} \cdot \frac{ee^T}{\|e\|^2}$; $V_+ = WV$

$x = x_+$; $V = V_+$

end for

Furthermore, we also tested Shor's R-algorithm with crucial coefficients $\frac{1}{3}$ and $\frac{1}{2}$ instead of $\frac{2}{3}$ in $W = I - \frac{2}{3}\frac{ee^T}{\gamma\|e\|^2}$ as well. The total number of iterations fluctuates a bit, but it is still not comparable with the SR1 and BFGS methods. Burke et al. [6] studied how Shor's R-algorithm behaves on convex quadratics, and showed the linear convergence of this algorithm under exact line search. In addition, they also performed a comprehensive experiment of this algorithm on such functions with different values of crucial coefficients.

In all, we conclude that secant methods like BFGS and SR1 learn curvature better than Shor.

4.6.2 Smooth Rosenbrock functions

The class of smooth Rosenbrock functions is a well-known test case in smooth optimization [36]. Here, based on comparison, it turns out that the BFGS method is better than the SR1 method in all aspects. At the same time, SR1 is better than Shor's R-algorithm.

Rosenbrock function

Define the Rosenbrock function to be

$$f(x) = \frac{1}{4}(x_1 - 1)^2 + \sum_{i=1}^{m-1}(x_{i+1} - x_i^2)^2.$$

The unique minimizer is $x^* = (1, 1, .., 1)$ with $f(x^*) = 0$.

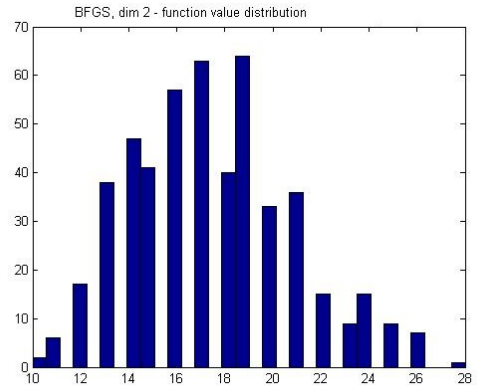
To make a systematic comparison, we set the coefficients of the Wolfe conditions $c_1 = 10^{-8}$ and $c_2 = 0.5$, the total number of function evaluations to be 3000, and the tolerance of termination to be 10^{-15}. Therefore, each empirical run will terminate because either it uses up all 3000 function evaluations or the function value is less than 10^{-15}. When it comes to the initial setting, the initial point of algorithms is either generated by a multivariate normal distribution $\mathcal{N}(0, 100)$ or chosen to be $(-1, 1, 1, .., 1)$, a point far from the minimizer (which lies in a "curved valley"); the initial Hessian is the identity matrix.

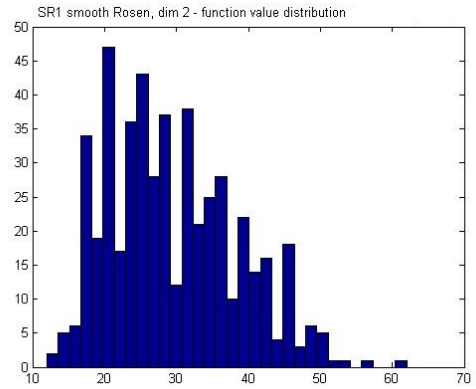
We explore $m = 2, 4, 8$ to compare these three algorithms numerically.

The BFGS vs SR1 method

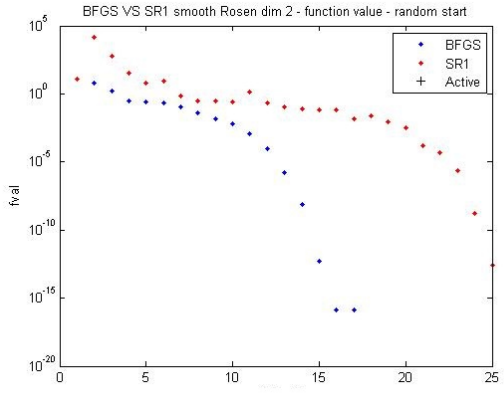
- Case: $m = 2$.

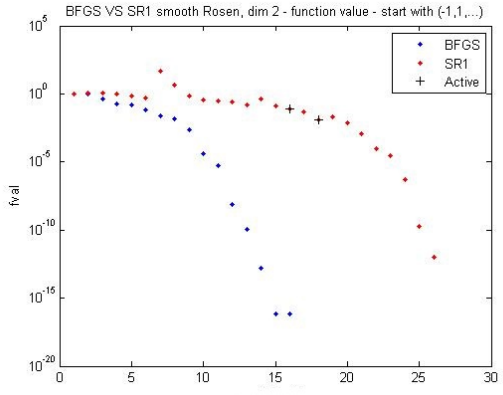
We apply both the BFGS and SR1 method to 500 randomly generated starting points, and plot the histograms of total number of function evaluations required to terminate. Notice that BFGS algorithm takes fewer function evaluations in general.





Two representative trajectories below show how the BFGS and SR1 methods behave with the particular hard starting point and a randomly generated one respectively. These two figures show the function values along all evaluations including those in the line search.

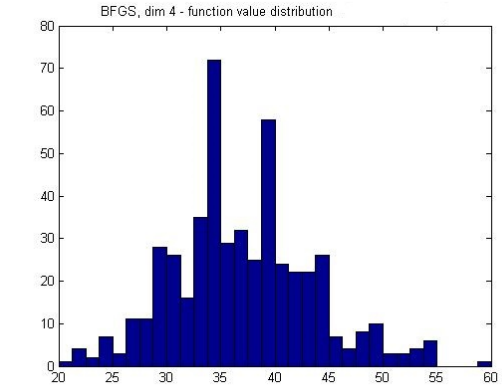


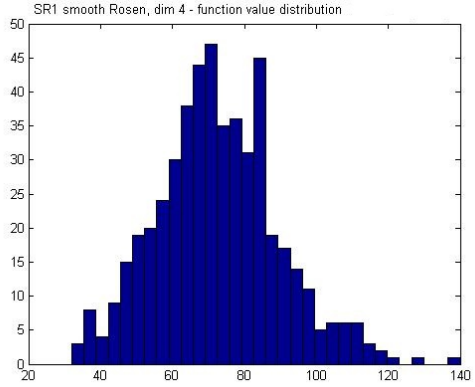


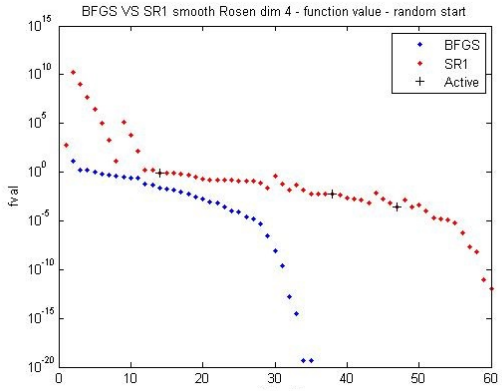
- Case: $m = 4$.

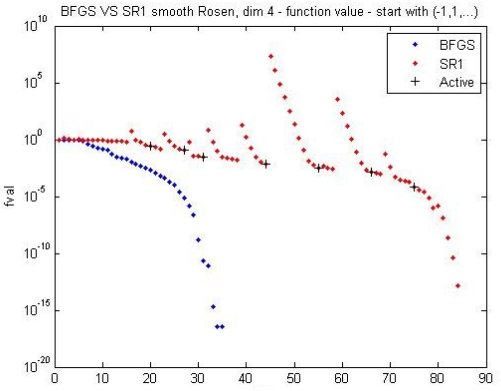
  With the same settings as $m = 2$, we have the following four figures. Even though the trajectory of the BFGS method seems to be monotonic, it is not actually. In fact, we can find the nonmonotonicity around 10th evaluations

in this plot.



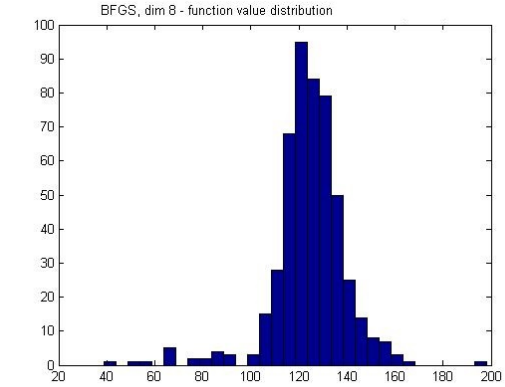


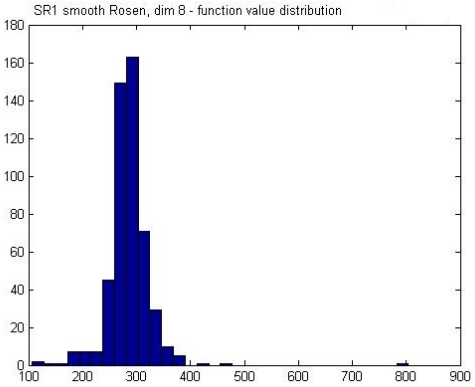


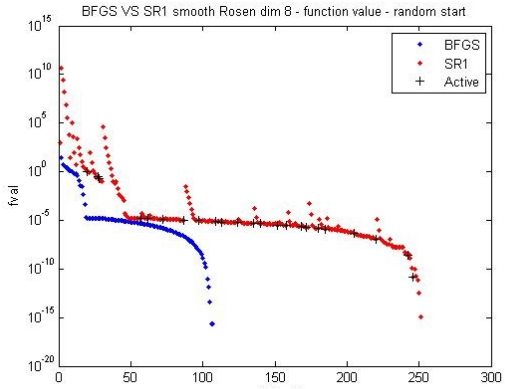


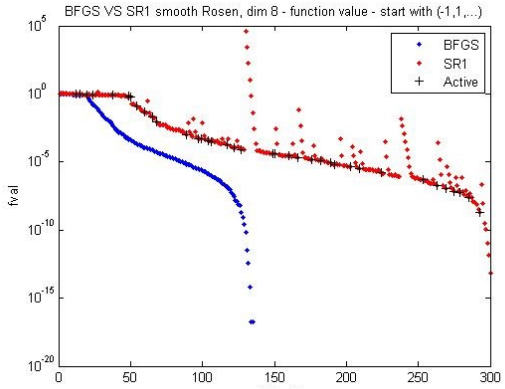
- Case: $m = 8$.

  With the same settings as $m = 2$, we have the following four figures.









.

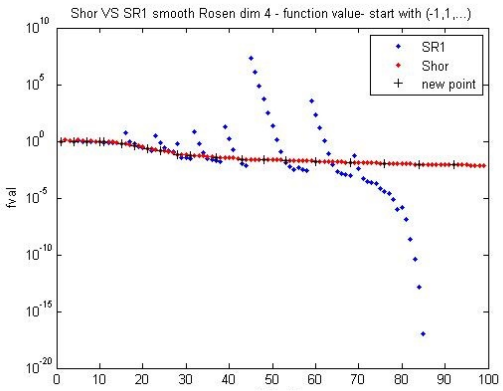
We can observe that the line search for SR1 may take a large number of iterations at certain points. On the other hand, the BFGS method is quite stable. In addition, the number of function values for the SR1 algorithm to terminate is often around twice that required by the BFGS algorithm on these examples.

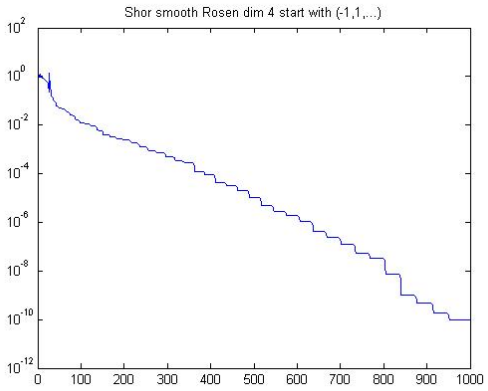
In all, SR1 is not learning the scale of good steps as well as BFGS.

The SR1 vs Shor's R-algorithm

Consider the same setting as above when $m = 4$. We again compare SR1 and Shor's R-algorithm with the particular hard starting point.

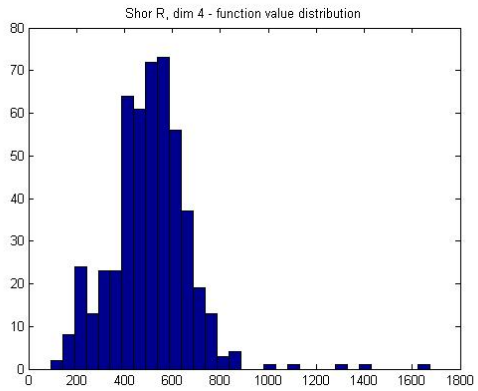
The left figure, a typical case, shows the number of evaluations required by the SR1 method is between 80 to 90. However, Shor's R-algorithm shows little progress within 100 iterations. The right figure indicates that Shor's R-algorithm terminates after a thousand evaluations.

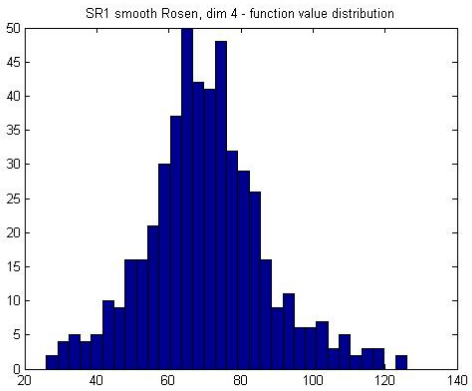




Moreover, we also test Shor's R-algorithm with the coefficients $\frac{1}{3}$ or $\frac{1}{2}$ instead of $\frac{2}{3}$ in $W = I - \frac{2}{3}\frac{ee^T}{\|e\|^2}$. It turns out that those modifications improve the result only in a limited way. Specifically, the total number of evaluations can be reduced to around seven hundred. Clearly, Shor's R-algorithm is still not as favorable as the SR1 method.

Given the undesirable performance of Shor's R-algorithm, we replace the threshold of termination from 10^{-15} to 10^{-8}. Here are two histograms of the number of evaluations required to reach this threshold given 500 randomly generated tests.





As we see, Shor's R-algorithm may takes 500 evaluations on average; however, SR1 only takes around 70. In all, we can safely draw the conclusion that

Shor's R-algorithm is much slower than the SR1 and BFGS methods in smooth cases. For the rest of the tests, we will focus on the comparison between SR1 and BFGS.

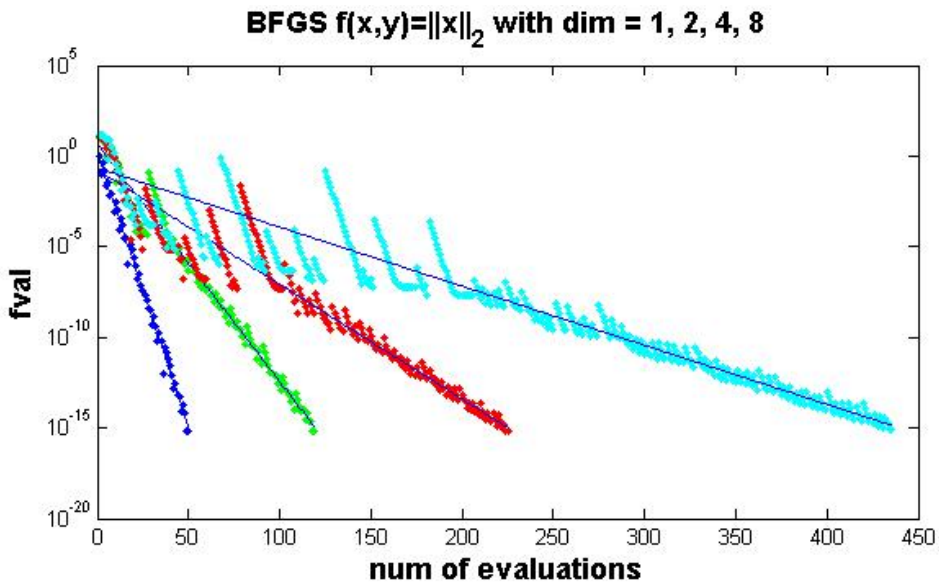
4.6.3 Norm function

The main purpose of this chapter is to see whether these algorithms can solve nonsmooth optimization problems in a reasonable manner. One of the easiest convex nonsmooth functions we can think of is the norm function. In addition, the proof of a convergence rate is known [23]. Therefore, the norm function is a good example to compare the BFGS and SR1 methods, and numerical results may provide us some intuition into the similarities and differences between the BFGS and various SR1 methods.

For this example, an interesting observation is that the SR1 method with absolute Hessian modification typically converges to the optimal solution with linear rate $r_{SR1} \approx 1 - \frac{2}{n}$ where n is the dimension of the problem. This rate of convergence is better than the BFGS method, where the rate $r_{BFGS} \approx 1 - \frac{1}{2n}$, although the $BFGS$ method is more stable than the SR1 method.

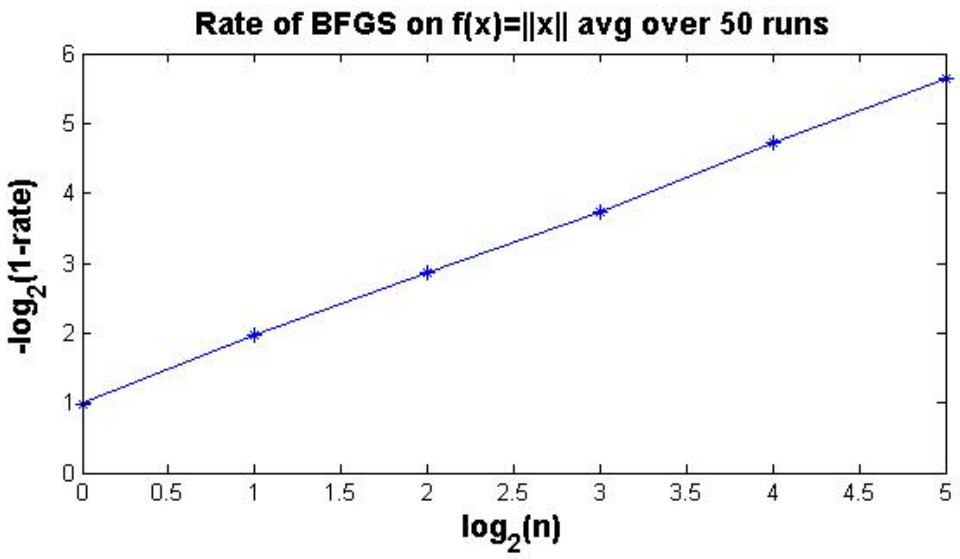
BFGS

To make a fair comparison, we apply the same settings as in the previous experiments. Note that the blue, green, red, and cyanide lines in the plot represent function evaluations for the BFGS method with dimension 1, 2, 4 and 8 respectively.



As we see, our results matches the previous observations by Lewis and Overton [23] that the BFGS method is seemingly linearly convergent for norm functions. In fact, they actually prove it for the dimension equal to 2. We can also see that the behavior seems to be reasonably stable near the optimal solution, but the line search is unstable at the beginning.

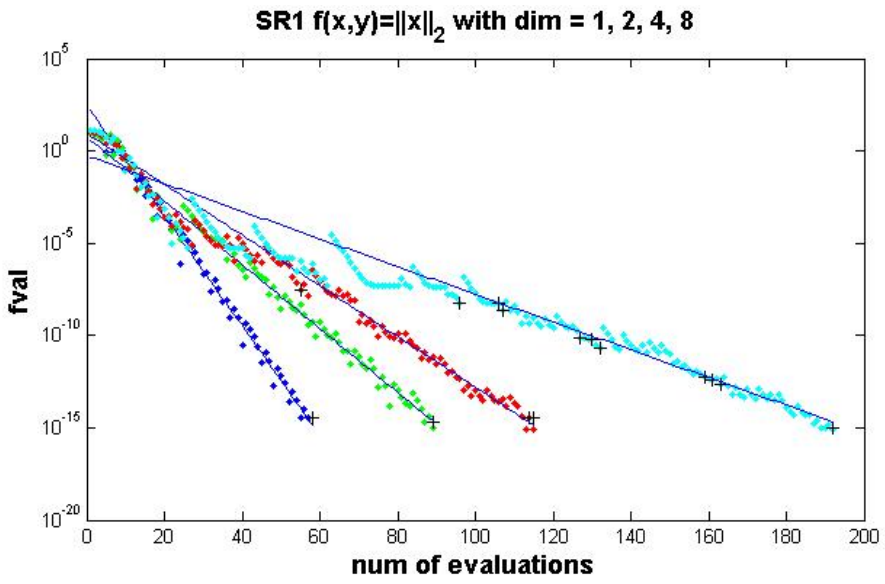
In addition, we extend the dimension up to 32, and have the following observation.



This plot underlies the average observed rate, namely the estimated Q-linear convergence rate for the sequence of function values. The relation between $-\log_2(1-rate)$ and $\log_2(n)$ is close to but a little better (lower) than the line $y = x+1$ as shown in the plot, meaning r_{BFGS} is slightly better than $1 - \frac{1}{2n}$ where n is the dimension of x.

SR1 with absolute Hessian

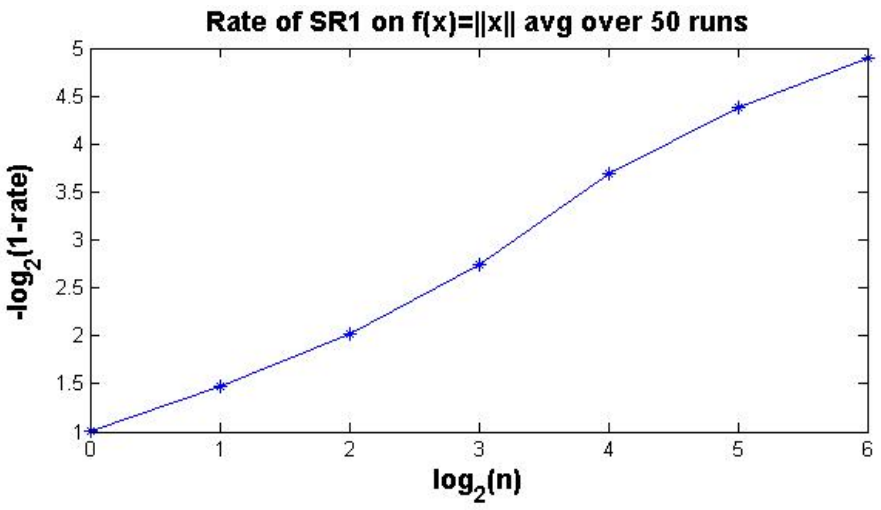
Given the same settings as before, we have the following plot for the SR1 method.



As we see, this plot is much better than that BFGS for in term of the number of total iterations. For example, BFGS requires around 450 function evaluations to converge whereas SR1 with absolute Hessian modification takes around 200 when $n = 8$.

However, the line search part for SR1 is less stable than for BFGS, around the optimal solution. The red line is a perfect example.

Furthermore, we also extend the dimension up to 32, and have the following plot.



This plot suggests the average observed a rate of SR1 method. The relation between $-\log_2(1 - rate)$ and $\log_2(n)$ is close to $y = \frac{2}{3}x + 1$, meaning $r_{SR1AbsHess}$ is approximately $1 - \frac{1}{2n^{\frac{2}{3}}}$.

Discussion

In summary, the experiments suggest that the SR1 update builds a useful local metric approximation, because it converges quite well. However, it is also true that SR1 is not as stable as the BFGS method. This does not surprise us intuitively, because rank-one updating cannot provide enough freedom to develop a matrix with all desired characteristics, whereas a rank-two correction has advantages. Moreover, modifications in the SR1 method, such as the idea of absolute Hessian modification, make the behavior unpredictable.

4.6.4 Max quadratic functions

Minimizing the pointwise max of a list of quadratic functions is a simple representative nonsmooth optimization problem. Specifically, the problem is formulated by

$$\min_{x} \max_{i=1,..,m} f_i(x) \tag{6.3}$$

where $f_i(x) = x^T H_i x + b_i^T x$ for $x \in \mathbf{R}^n$.

If no property of convexity is enforced for each quadratic function, it becomes a nonconvex nonsmooth optimization problem. We could formulate this problem as the nonlinear program $\min\{t : f_i(x) \leq t, \ \forall \ i\}$ to seek a local solution, although such type of problems are not easy generally, and in particular are NP-hard to solve globally. Here we are just interested in the problem (6.3) as a simple test case.

In order to find a reasonable criterion of termination, all test functions we generate have 0 the unique minimizer. In particular, we sample each $g_i \in \mathbf{R}^n$ and each symmetric matrix $H_i = A_i + A_i^T$ for $i = 1, .., m-1$ randomly. In addition, we set $g_m = -\sum_{i=1}^{m-1} g_i$, and $H_m = (1-\lambda)I$ where $\lambda = \lambda_{min}(\sum_{i=1}^{m-1} H_i)$.

To make a consistent comparison, we choose $c_1 = 10^{-8}$ and $c_2 = 0.5$ as the coefficients of Wolfe condition, maximal 3000 function evaluations, and the tolerance of termination to be 10^{-8}. Therefore, the algorithm will terminate if either it runs out of all 3000 function evaluations or the function value reaches the tolerance 10^{-8}. Moreover, the initial point of algorithms is sampled from a multivariate normal distribution $\mathcal{N}(0, 100)$, and initial Hessian is the identity matrix.

Lastly, we choose the dimension of the problem to be $n = 5$ for all numerical

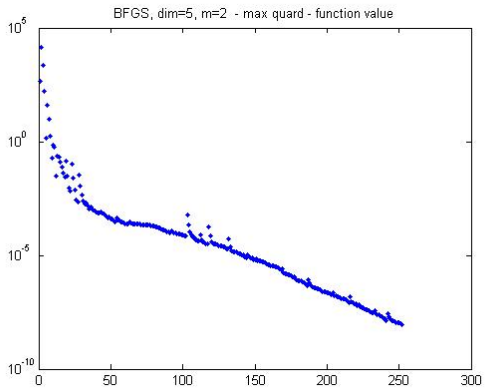
experiments, and the value of m may vary.

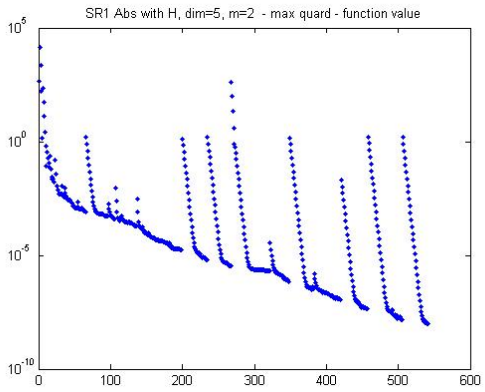
The BFGS vs SR1 method

In this part, we test the BFGS and SR1 methods for the same randomly generated functions and starting point simultaneously, and compare the result by sample trajectories and histograms.

- Case: $m = 2$.

  The left trajectory shows how the BFGS method behaves and the right trajectory shows how the SR1 method with absolute Hessian modification works given the same starting point.

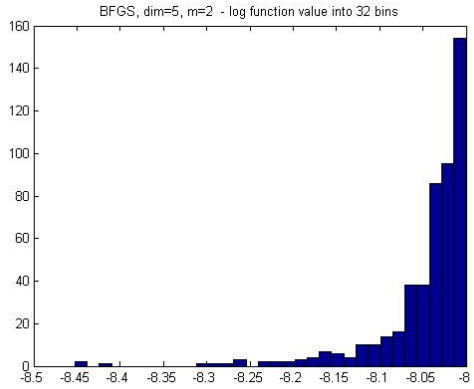


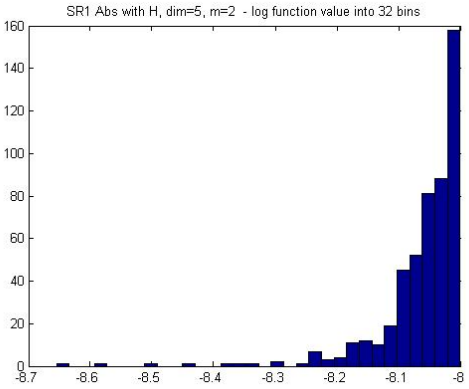


  As we see, both BFGS and SR1 can successfully reach the tolerance of termination. BFGS requires around 260 number of function evaluations; whereas SR1 requires around 570. The BFGS algorithm is reasonably stable, but the performance of SR1 is much less so at some specific points.

  In addition to sample trajectories, we also run both algorithms with 500 randomly generated starting points. The plot left below shows the distribution of function values of the BFGS method at termination in the log

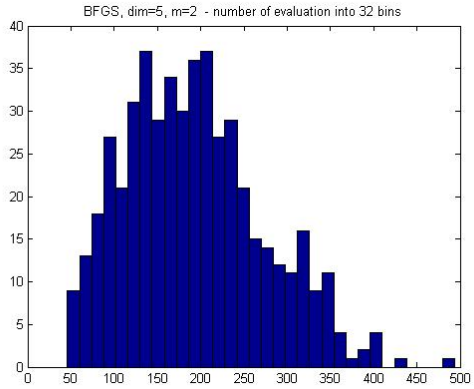
base 10 scale; the plot right below shows that of the SR1 method.

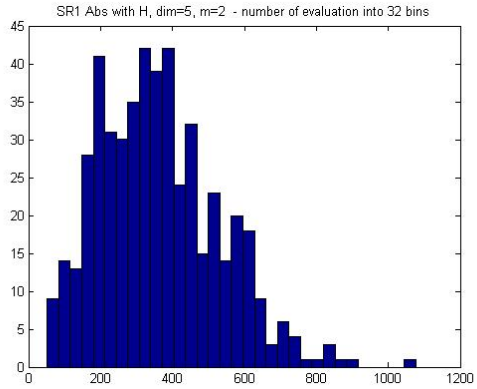




Form these two plots, we learn that both BFGS and SR1 successfully reach the toleration 10^{-8} for all examples. This implies the success of both algorithms.

Another interesting aspect of the experiments above is the total number of evaluations at termination. The left plot below shows the distribution of the number of evaluations for the BFGS method; the right plot below shows that for the SR1 method.





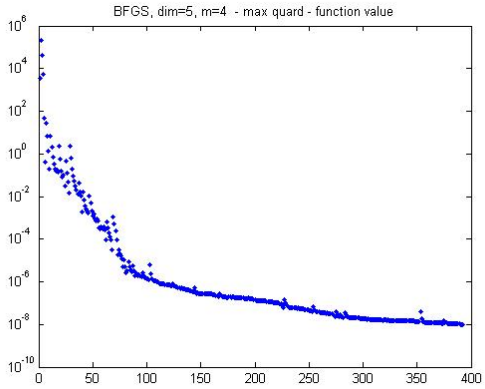
We can observe that most numbers of evaluations required to reach tolerance are around 100 - 300 and 200 - 800 for the BFGS and SR1 algorithms respectively. The plots of a single trajectory illustrate this result.

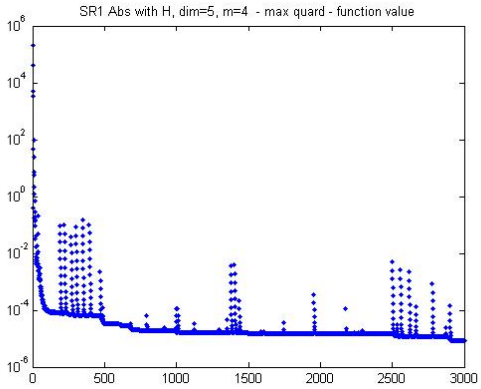
Experiments with $m = 3$ are similar to the case with $m = 2$, so we skip

them.

- Case: $m = 4$.

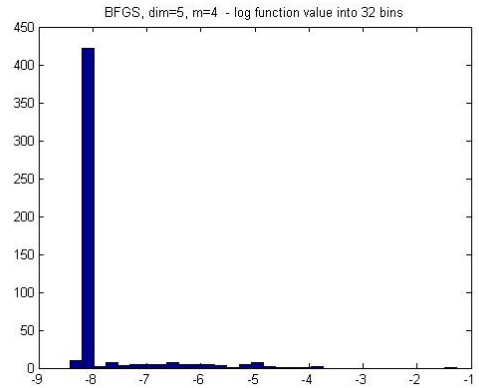
  The left trajectory shows how the BFGS method typically behaves and the right trajectory shows how the SR1 method with absolute Hessian Hessian typically works given the same starting point.

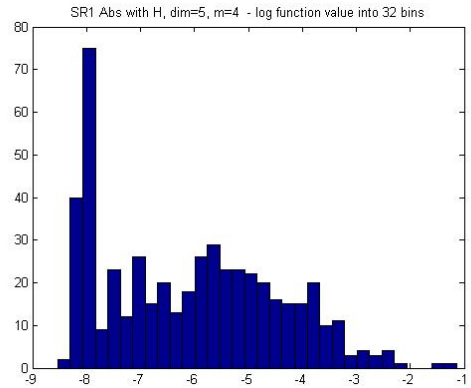




As we see, BFGS requires around 400 function evaluations to terminate; whereas SR1 can only reach 10^{-5} after using up all 3000 evaluations. Clearly, the performance of the line search in SR1 is not as stable.

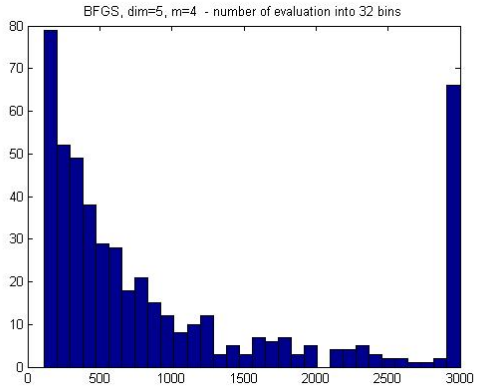
We also run both algorithms with 500 randomly generated starting points. The plot left below shows the distribution of function values of the BFGS method at termination in the log base 10 scale; the plot right below shows that of the SR1 method.

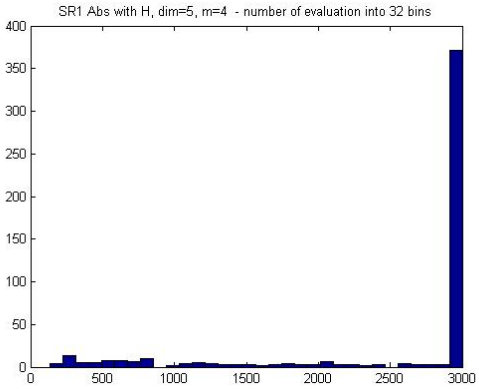




From these two plots, we learn that almost all BFGS runs terminate successfully, while only around 30 percent of SR1 runs do so. This indicates the power of the the BFGS method when problems become harder.

The left plot below shows the distribution of the total number of evaluations at termination for the BFGS method; the right plot below shows that for the SR1 method.





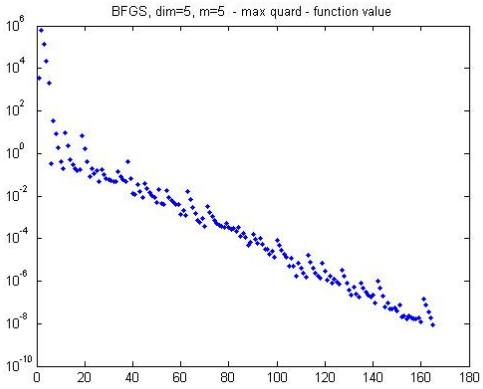
When it comes to numbers of evaluations required to reach tolerance, more than half of them are less than 1500 for the BFGS algorithm, much better than SR1 method. The plots of a single trajectory also illustrate this result again.

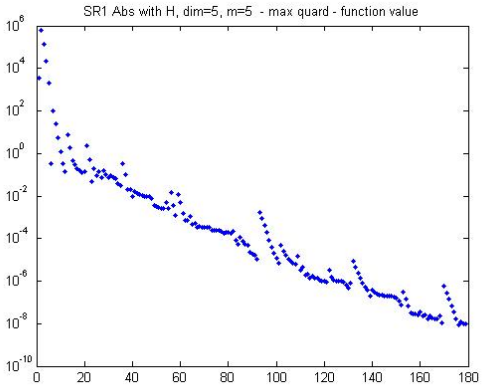
Experiments with $m = 5$ are similar.

- Case: $m = 6$.

  Note that the minimizer here is sharp, like unique solutions in linear programming. This may possibly explain different behavior we observe.

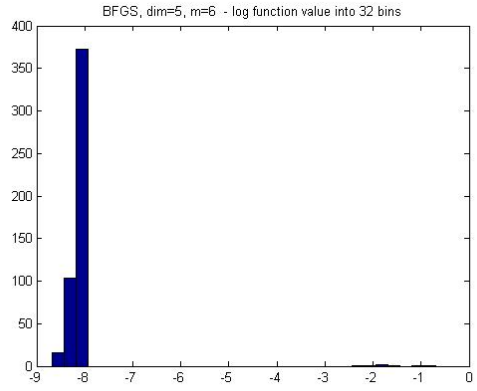
  The left trajectory shows how the BFGS method typically behaves and the right trajectory shows how the SR1 method with absolute Hessian modification works given the same starting point.

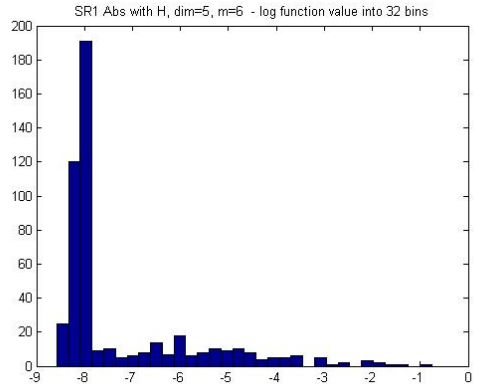




From these plots, we can observe that the behavior of BFGS and SR1 are similar for this problem.

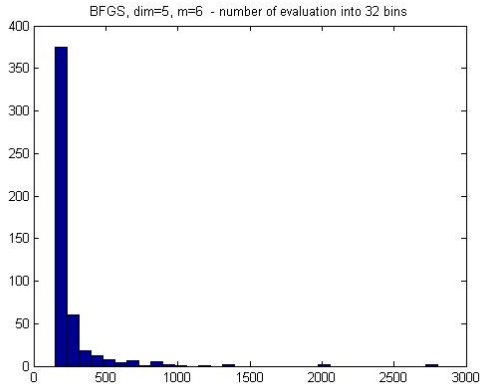
In addition, the distribution of the function values required for the BFGS method and that of the SR1 method for termination (in log base 10 scale) are presented on the left and right respectively.

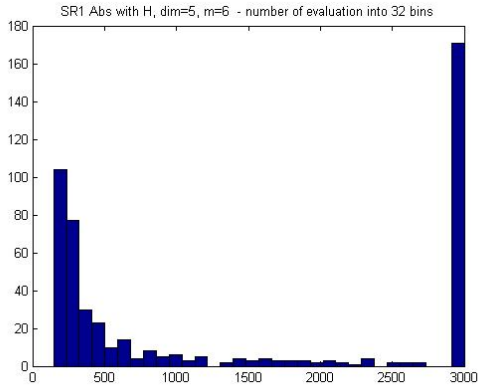




From these two plots, we learn that almost all BFGS runs terminates successfully, while only a half of SR1 runs do so. Thus, the BFGS method performs better than SR1 overall.

The left plot below shows the distribution of the total number of function evaluations at termination for the BFGS method; the right plot below shows that for the SR1 method.



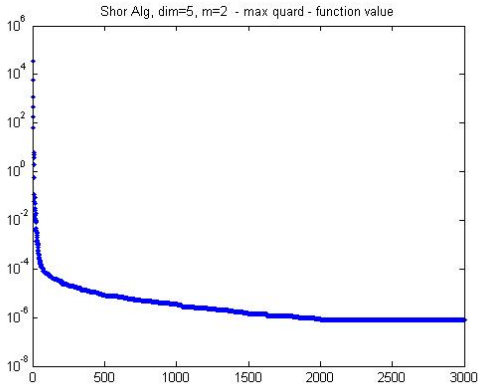


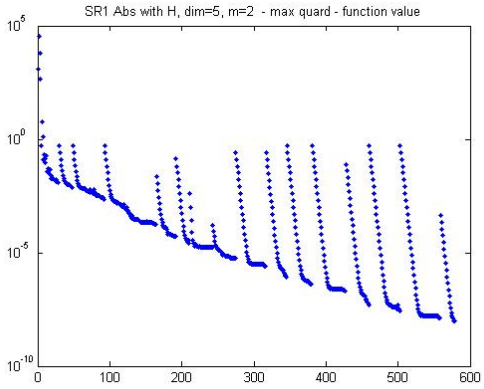
If we consider numbers of evaluations required to reach tolerance, more than 70 percent are less than 500 for BFGS algorithm, much better than the SR1 method again.

The SR1 vs Shor's R-algorithm

We can apply the same tests to Shor's R-algorithm, and compare it with the SR1 method. In general, Shor's R-algorithm is much slower than the SR1 algorithm. To make it clear, we present the same example with $m = 2$.

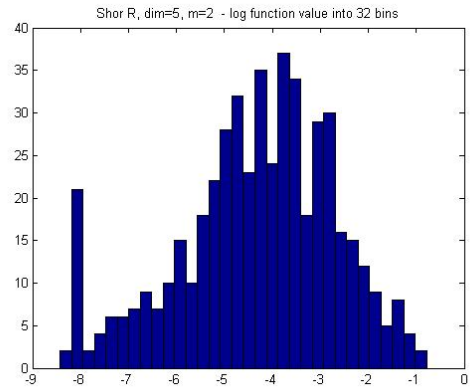
The left plot below shows how Shor's R-method behaves and the right plot below shows how the SR1 method with absolute Hessian modification works with the same starting point.

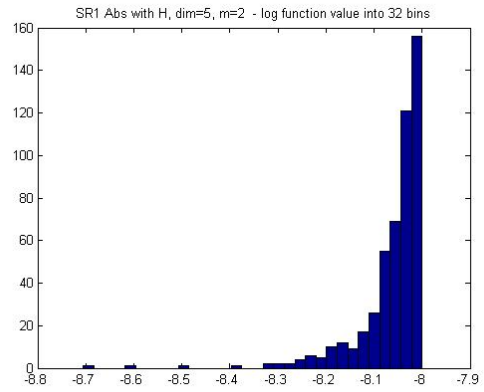




As we see, SR1 requires almost 600 numbers of function evaluations to terminate; whereas Shor's R-algorithm can only reach 10^{-6} after using up all 3000 evaluations, even though the performance of line search in SR1 is very unstable.

In addition to sample trajectories, we also run both algorithms with 500 randomly generated starting points. The left plot below shows the distribution of function values of Shor's R-method in the log base 10 scale at termination; the right plot below shows that of SR1 method.

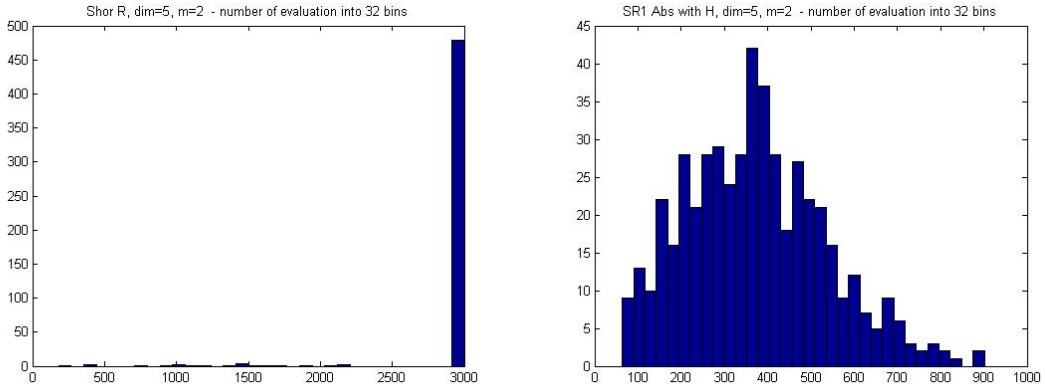




All SR1 runs reach the desired tolerance 10^{-8}, much better than Shor's R-method.

Regarding the total number of evaluations at termination, the left plot below

shows the distribution of the total number of evaluations for Shor's R-method; the right plot below shows that for the SR1 method.



As we see, more than 90 percent of Shor runs uses up all 3000 function evaluations budgeted; however, the max budget required by SR1 method is around 900.

BIBLIOGRAPHY

[1] A. ASL AND M. OVERTON, *Analysis of the gradient method with an Armijo-Wolfe line search on a class of nonsmooth convex functions*, arXiv:1711.08517, (2017).

[2] R. G. BLAND, D. GOLDFARB, AND M. J. TODD, *The ellipsoid method: A survey*, Operations Research, 29 (1981), pp. 1039–1091.

[3] J. BORWEIN AND A. LEWIS, *Convex Analysis and Nonlinear Optimization.*, Number 3 in CMS Books in Mathematics, Springer-Verlag, 2000.

[4] L. X. BOYD, STEPHEN AND A. MUTAPCIC, *Subgradient methods*, lecture notes of EE392o, Stanford University, Autumn Quarter 2004 (2003): 2004-2005, (2005).

[5] S. BUBECK, *Convex Optimization: Algorithms and complexity*, Trends Mach. Learn., 8 (2015), pp. 231–357.

[6] J. V. BURKE, A. S. LEWIS, AND M. L. OVERTON, *The speed of Shor's R-algorithm*, IMA Journal of Numerical Analysis, 28 (2008), pp. 711–720.

[7] BURKE, J.V., A. S. LEWIS, AND M. L. OVERTON, *A robust gradient sampling algorithm for nonsmooth, nonconvex optimization.*, SIAM J. Optim. 15, 751–779, (2005).

[8] I. D. C. D. BYATT AND C. J. PRICE., *Performance of various BFGS implementations with limited precision second-order information*, ANZIAM J., 45(4):511-522, (2004).

[9] Y.H. DAI, *A perfect example for the BFGS method*, Mathematical Programming, (2013).

[10] J. DENNIS AND J. MORE, *Quasi-Newton methods, motivation and theory*, SIAM review, 19(1): 46-89, (1977).

[11] L. DIXON, *Quasi newton techniques generate identical points II: The proofs of four new theorems*, Math. Program., 3 (1972), pp. 345–358.

[12] J. DUNAGAN AND S. VEMPALA, *A simple polynomial-time rescaling algorithm for solving linear programs*, In Proceedings of the 36th Annual ACM Symposium on Theory of Computing, pages 315-320. ACM,(2004).

[13] R. FLETCHER, *Practical Methods of Optimization; (2nd Ed.)*, Wiley-Interscience, 1987.

[14] R. FREUND, *Complexity of convex optimization using geometry-based measures and a reference point*, Math. Program., 99(2, Ser. A):197-221, (2004).

[15] J. GILBERT AND J. NOCEDAL, *Global convergence properties of conjugate gradient methods for optimization.*, SIAM J. Optim., 2: 21-42, (1992).

[16] A. GRIEWANK, *Broyden updating, the good and the bad!*, Doc. Math., Extra Volume (Optimization stories): 301-315, (2012).

[17] J. GUO AND A. LEWIS, *BFGS convergence to nonsmooth minimizers of convex functions*, arXiv: 1703.06690, SIAM J. Optim to appear, (2018).

[18] J. GUO AND A. LEWIS, *Rescaling nonsmooth optimization using BFGS and Shor updates*, arXiv: 1802.06453, submitted to Computational Optimization and Applications, (2018).

[19] J.-B. HIRIART-URRUTY AND C. LEMARECHAL, *Convex Analysis and Minimization Algorithms I*, Springer-Verlag, Berlin, 1993.

[20] D. C. S. JORGE J. MORE, *On the use of directions of negative curvature in a modified newton method*, Math. Program. 16 (1979) 1-20, (1977).

[21] F. KAPPEL AND A. KUNTSEVICH., *An implementation of Shor's R-algorithm.*, Comput. Optim. Appl., 15(2): 193-205, (2000).

[22] K. KIWIEL, *Methods of descent for nondifferentiable optimization.*, Lecture Notes in Mathematics, vol. 1133. Springer, Berlin, (1985).

[23] A. S. LEWIS AND M. L. OVERTON, *Nonsmooth optimization via quasi-Newton methods*, Math. Program. 141.1-2: 135-163, (2013).

[24] A. S. LEWIS AND S. ZHANG, *Nonsmoothness and the BFGS method*, Journal of Optimization Theory and Applications, 165 (2015), 151-171. (2015).

[25] Z. LUO AND P. TSENG, *On the convergence of the coordinate descent method for convex differentiable minimization*, J. Optim. Theory Appl. 72(1): 7-35, (1992).

[26] W. MASCARENHAS, *The BFGS method with exact line searches fails for non-convex objective functions*, Math. Program., 99: 49-61, (2004).

[27] Y. NESTEROV AND A. NEMIROVSKII, *Interior-point polynomial algorithms in convex programming*, volume 13 of SIAM Studies in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, (1994).

[28] J. NOCEDAL AND S. J. WRIGHT, *Numerical Optimization*, Springer, New York, 2nd ed., 2006.

[29] A. B. J. NOVIKOFF, *On convergence proofs on perceptrons*, Proceedings of the Symposium on the Mathematical Theory of Automata, 12 (1962), pp. 615–622.

[30] F. OZTOPRAK AND S. I. BIRBIL, *A symmetric rank-one quasi-Newton line-search method using negative curvature directions*, Optimization Methods and Software, 26 (2011), pp. 455–486.

[31] M. POWELL, *On the convergence of the DFP algorithm for unconstrained optimization when there are only two variables*, Math. Program., 87:281-301, (2000).

[32] M. J. D. POWELL, *Some global convergence properties of a variable metric algorithm for minimization without exact line searches*, In Nonlinear Programming, Amer. Math. Sco. SIAM-AMS Proc., IX (1976), pp. 53–72.

[33] M. J. D. POWELL., *Updating conjugate directions by the BFGS formula*, Math. Programming, 38(1):29-46, (1987).

[34] H. F. K. R. H. BYRD AND R. B. SCHNABEL., *Analysis of a symmetric rank-one trust region method*, SIAM Journal on Optimization, 6(4): 1025–1039, (1996).

[35] F. ROSENBLATT, *The perceptron: A probabilistic model for information storage and organization in the brain*, Psychological Review, 65, 386-408, (1958).

[36] H. ROSENBROCK, *An automatic method for finding the greatest or least value of a function*, The Computer Journal. 3 (3): 175-184, (1960).

[37] H. SENDOV., *Nonsmooth analysis of lorentz invariant functions*, SIAM J. Optim. 18(3): 1106-1127, (2007).

[38] N. Z. SHOR, *Minimization methods for nondifferentiable functions*, volume 3 of Springer Series in Computational Mathematics, (1985).

[39] M. YAN, *Extension of convex function*, J. Convex Anal., 21(4):965-987, (2014).